



Messaging Framework API

Version 1.0
March 6, 2008

**PLEASE REVIEW THE TERMS OF THE FOUNDATION API LICENSE VERSION 1.0
BEFORE READING THIS DOCUMENT.**

You may view the Foundation API License v1.0 at:

http://www.limofoundation.org/images/stories/pdf/limo_foundation_api_license_v1-0.pdf

This is an agreement between you ("You") and LiMo Foundation related to the use of the APIs contained within this document. By accessing, downloading, reading or otherwise using the APIs contained within this document, You accept the terms of the Foundation API License v1.0. If You do not agree to the terms of this agreement, do not download, read and/or otherwise use any of the APIs.

If You are agreeing to be bound by the Foundation API License v1.0 on behalf of your employer, You represent and warrant that you have full legal authority to bind your employer to this agreement. If You do not have the requisite authority, You may not download, read and/or otherwise use any of the APIs on behalf of your employer.

NOTICES

This document and the technical information it contains is subject to change by LiMo Foundation and its members at any time without notice as authorized by LiMo Foundation bylaws and other documents governing LiMo Foundation, including amendments thereto. All statements regarding LiMo's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Any reference in this document to non-LiMo Foundation Web sites do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of this document or any product described in this document and use of those Web sites is at your own risk.

Information concerning non-LiMo products was obtained from the suppliers of those products or a public announcement or source. LiMo cannot confirm the accuracy of that information or the performance, compatibility or any other claims related to non-LiMo products.

All names or other identifying information used in examples in this document are fictitious and any similarity to the names or other identifying information of an actual person or organization is entirely coincidental.

THIS DOCUMENT AND ALL INFORMATION IN THIS DOCUMENT ARE PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OR CONDITION, INCLUDING, BUT NOT LIMITED TO, ANY WARRANTY OR CONDITION OF ACCURACY OR COMPLETENESS, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR AGAINST INFRINGEMENT. ALL EXPRESS AND IMPLIED STATUTORY WARRANTIES ARE DISCLAIMED, INCLUDING WITHOUT LIMITATION, THE WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE OR NON-INFRINGEMENT.

Contents

1	Messaging Framework API	1
1.1	References	1
1.2	Overview	1
1.3	Industry Standard	1
1.4	External API Documents	1
1.5	Plug-in Extension Point Interface	2
1.6	Other Interfaces	2
1.7	Other Notes	2
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Page Index	7
4.1	Related Pages	7
5	Data Structure Documentation	9
5.1	_MSG_ACCOUNT_ID_LIST_S Struct Reference	9
5.2	_MSG_ACCOUNT_S Struct Reference	10
5.3	_MSG_CONTACT_LIST_S Struct Reference	12
5.4	_MSG_CONTACT_S Struct Reference	13
5.5	_MSG_FILTER_ID_LIST_S Struct Reference	15
5.6	_MSG_FILTER_LIST_S Struct Reference	16
5.7	_MSG_FILTER_S Struct Reference	17
5.8	_MSG_FOLDER_ID_LIST_S Struct Reference	19
5.9	_MSG_FOLDER_S Struct Reference	20
5.10	_MSG_FW_STORAGE_S Struct Reference	22
5.11	_MSG_MESSAGE_ID_LIST_S Struct Reference	26

5.12	_MSG_MESSAGE_REQUEST_S Struct Reference	27
5.13	_MSG_MESSAGE_S Struct Reference	29
5.14	_MSG_MESSAGE_STATUS_S Struct Reference	33
5.15	_MSG_MESSAGE_TYPE_LIST_S Struct Reference	35
5.16	_MSG_MESSAGE_TYPE_S Struct Reference	36
5.17	_MSG_PLG_STORAGE_S Struct Reference	37
5.18	_MSG_PLUGIN_FILTER_MANAGER_S Struct Reference	39
5.19	_MSG_PLUGIN_TRANSPORT_S Struct Reference	40
5.20	_MSG_QUERY_ID_LIST_S Struct Reference	41
5.21	_MSG_QUERY_LIST_S Struct Reference	42
5.22	_MSG_QUERY_S Struct Reference	43
5.23	_MSG_SORT_RULE_S Struct Reference	44
5.24	_MSG_SORT_TYPE_LIST_S Struct Reference	45
5.25	_MSG_STORAGE_ID_LIST_S Struct Reference	46
5.26	_MSG_STRING_S Struct Reference	47
5.27	_MSG_TRANSPORT_LISTENER_S Struct Reference	48
6	File Documentation	49
6.1	ComponentOverview.h File Reference	49
6.2	Msg_Framework_Storage_Interface.h File Reference	50
6.3	Msg_Mapi_Account.h File Reference	79
6.4	Msg_Mapi_Contact.h File Reference	80
6.5	Msg_Mapi_FilterRule.h File Reference	81
6.6	Msg_Mapi_Folder.h File Reference	84
6.7	Msg_Mapi_Message.h File Reference	85
6.8	Msg_Mapi_Message_Request.h File Reference	86
6.9	Msg_Mapi_Message_Status.h File Reference	87
6.10	Msg_Mapi_QueryRule.h File Reference	88
6.11	Msg_Mapi_Types.h File Reference	91
6.12	Msg_Plugin_Filter_Interface.h File Reference	104
6.13	Msg_Plugin_Storage_Interface.h File Reference	107
6.14	Msg_Plugin_Transport_Interface.h File Reference	121
6.15	Msg_Plugin_Types.h File Reference	128
7	Page Documentation	131
7.1	Overview of the LiMo Messaging Plug-ins	131
7.2	Sequence Charts for the LiMo Messaging API	138

7.3 Filter Plug-in: Set Filter List to Filter Messages	139
--	-----

Chapter 1

Messaging Framework API

1.1 References

[1] *Software Architecture Document*, Version 1.1 Revision 1.4, LiMo Foundation, 11 September 2007

[2] *Messaging API*, Version 1.0, LiMo Foundation, 1 February 2008

1.2 Overview

The LiMo Messaging Framework provides services for applications to send, receive, parse and assemble messages over arbitrary message delivery mechanisms such as MMS, SMS, email, and i-mode message. The messaging services consist of common functions and sets of message delivery service-specific functions. Also, the Messaging Framework provides a collection of data structures to implement the services.

The Messaging Framework is built on a modular, plug-in architecture for the flexibility and extendability of the messaging system. The functional logic for a particular service type resides in a plug-in module, not in the Messaging Framework, allowing the messaging system to be customized for a particular use-case. The Messaging Framework provides the interface and protocols for plug-ins and applications to communicate with each other in a consistent manner, thus providing simplified messaging for applications.

In addition, the Messaging Framework also manages the plug-ins, providing plug-in discovery and initialization as well as other full life-cycle management. Therefore, plug-ins are also required to provide managerial functions and structures to cooperate with the Framework. For more information, see [Overview of the LiMo Messaging Plug-ins](#) for more information.

1.3 Industry Standard

None

1.4 External API Documents

None

1.5 Plug-in Extension Point Interface

None

1.6 Other Interfaces

None

1.7 Other Notes

None

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

_MSG_ACCOUNT_ID_LIST_S (Represents an account id list)	9
_MSG_ACCOUNT_S (Represents an account)	10
_MSG_CONTACT_LIST_S (Represents a contact list)	12
_MSG_CONTACT_S (Represents a contact)	13
_MSG_FILTER_ID_LIST_S (Represents a filter id list)	15
_MSG_FILTER_LIST_S (Represents a filter list)	16
_MSG_FILTER_S (Represents a filter)	17
_MSG_FOLDER_ID_LIST_S (Represents a folder id list)	19
_MSG_FOLDER_S (Represents a folder)	20
_MSG_FW_STORAGE_S (Represents a framework storage plugin handle)	22
_MSG_MESSAGE_ID_LIST_S (Represents a message id list)	26
_MSG_MESSAGE_REQUEST_S (Represents a request that will be sent to messaging frame- work)	27
_MSG_MESSAGE_S (Represents a message)	29
_MSG_MESSAGE_STATUS_S (Represents a message status)	33
_MSG_MESSAGE_TYPE_LIST_S (Represents a message types list)	35
_MSG_MESSAGE_TYPE_S (Represents a message type)	36
_MSG_PLG_STORAGE_S (Represents a message storage plugin handle)	37
_MSG_PLUGIN_FILTER_MANAGER_S (Represents a filter manager)	39
_MSG_PLUGIN_TRANSPORT_S (Represents a transport plugin handle)	40
_MSG_QUERY_ID_LIST_S (Represents a query Id list)	41
_MSG_QUERY_LIST_S (Represents a query list)	42
_MSG_QUERY_S (Represents a query rule)	43
_MSG_SORT_RULE_S (Represents a sort rule)	44
_MSG_SORT_TYPE_LIST_S (Represents a sort type list)	45
_MSG_STORAGE_ID_LIST_S (Represents a storage id list)	46
_MSG_STRING_S (Represents a string)	47
_MSG_TRANSPORT_LISTENER_S (Represents a transport listener handle)	48

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

ComponentOverview.h (Component Programming Interface Overview Header Template for Doxygen)	49
Msg_Framework_Storage_Interface.h (Defines the interfaces that framework storage plugin MUST implement)	50
Msg_Mapi_Account.h (Defines the account structure)	79
Msg_Mapi_Contact.h (Defines the contact structure)	80
Msg_Mapi_FilterRule.h (Defines the filter related structures)	81
Msg_Mapi_Folder.h (Defines the folder related structures)	84
Msg_Mapi_Message.h (Defines the message structure)	85
Msg_Mapi_Message_Request.h (Defines the message request structure)	86
Msg_Mapi_Message_Status.h (Defines the message status structure)	87
Msg_Mapi_QueryRule.h (Defines the query related structure)	88
Msg_Mapi_Types.h (Defines general types)	91
Msg_Plugin_Filter_Interface.h (Defines the filter interfaces that filter plugin MUST implement)	104
Msg_Plugin_Storage_Interface.h (Defines the interfaces that message storage plugin MUST implement)	107
Msg_Plugin_Transport_Interface.h (Defines the transport interfaces that transport plugin MUST implement)	121
Msg_Plugin_Types.h (Defines general types for plugins)	128

Chapter 4

Page Index

4.1 Related Pages

Here is a list of all related documentation pages:

Overview of the LiMo Messaging Plug-ins	131
Sequence Charts for the LiMo Messaging API	138

Chapter 5

Data Structure Documentation

5.1 _MSG_ACCOUNT_ID_LIST_S Struct Reference

```
#include <Msg_Mapi_Types.h>
```

5.1.1 Detailed Description

Represents an account id list.

Data Fields

- [int iCount](#)
- [MSG_ACCOUNT_ID_T * accountIds](#)

5.1.2 Field Documentation

5.1.2.1 int _MSG_ACCOUNT_ID_LIST_S::iCount

The count of message ids

5.1.2.2 MSG_ACCOUNT_ID_T* _MSG_ACCOUNT_ID_LIST_S::accountIds

The pointer to the account id list

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Types.h](#)

5.2 _MSG_ACCOUNT_S Struct Reference

```
#include <Msg_Mapi_Account.h>
```

5.2.1 Detailed Description

Represents an account.

All the fields except for `pExData` contain common information for an account. The field `pExData` points to the user customized structure. This customized structure should be used by application and message storage plugin.

For the more detailed information of account, please refer to `OverviewStructureAccount`. This structure is extendable. Please refer to *Extendability* (see *Messaging API* [2]).

Data Fields

- [MSG_ACCOUNT_ID_T](#) `accountId`
- [MSG_STRING_S](#) `accountName`
- [MSG_MESSAGE_TYPE_S](#) `msgType`
- [MSG_FOLDER_ID_T](#) `rootFolderId`
- [MSG_STORAGE_ID_T](#) `storageId`
- [MSG_STORAGE_ID_T](#) `defaultMsgStorageId`
- `size_t` `extDataSize`
- `void *` `pExtData`

5.2.2 Field Documentation

5.2.2.1 MSG_ACCOUNT_ID_T _MSG_ACCOUNT_S::accountId

Indicates the unique account index.

5.2.2.2 MSG_STRING_S _MSG_ACCOUNT_S::accountName

Indicates the display name for this account.

5.2.2.3 MSG_MESSAGE_TYPE_S _MSG_ACCOUNT_S::msgType

Indicates the message type of an account, such as SMS, MMS, EMAIL and i-mode message.

5.2.2.4 MSG_FOLDER_ID_T _MSG_ACCOUNT_S::rootFolderId

Indicates the root folder which is assigned to this account.

5.2.2.5 MSG_STORAGE_ID_T _MSG_ACCOUNT_S::storageId

Indicates the storage id in which this account (metadata and extension data) is stored.

5.2.2.6 MSG_STORAGE_ID_T _MSG_ACCOUNT_S::defaultMsgStorageId

Indicates default storage id for the message of this account

5.2.2.7 size_t _MSG_ACCOUNT_S::extDataSize

Indicates the size of pExtData

5.2.2.8 void* _MSG_ACCOUNT_S::pExtData

Used to add account extension data. For example, the mail box setting of Email account.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Account.h](#)

5.3 `_MSG_CONTACT_LIST_S` Struct Reference

```
#include <Msg_Mapi_Contact.h>
```

5.3.1 Detailed Description

Represents a contact list.

Data Fields

- `int iCount`
- `MSG_CONTACT_S * pContacts`

5.3.2 Field Documentation

5.3.2.1 `int _MSG_CONTACT_LIST_S::iCount`

Indicates the number of the contacts in the list.

5.3.2.2 `MSG_CONTACT_S* _MSG_CONTACT_LIST_S::pContacts`

The pointer to the contact list.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Contact.h](#)

5.4 _MSG_CONTACT_S Struct Reference

```
#include <Msg_Mapi_Contact.h>
```

5.4.1 Detailed Description

Represents a contact.

This structure includes the contact basic information. It can present various contact type such as telephone number, email address and so on.

This structure is extendable. Please refer to Extendability (see *Messaging API* [2]) .

Data Fields

- [MSG_CONTACT_TYPE_T](#) `contactType`
- unsigned char * `contactVal`
- [MSG_STRING_S](#) `contactName`
- size_t `extDataSize`
- void * `pExtData`

5.4.2 Field Documentation

5.4.2.1 MSG_CONTACT_TYPE_T _MSG_CONTACT_S::contactType

Indicates contact type.

Two types are predefined:

[MSG_CONTACT_TYPE_EMAIL](#) is for email address like `abc@example.com`

[MSG_CONTACT_TYPE_PLMN](#) is for phone number like `+10123456789`

5.4.2.2 unsigned char* _MSG_CONTACT_S::contactVal

The value of the contact. It should be encoded in ASCII.

For example, the real phone number or email address.

5.4.2.3 MSG_STRING_S _MSG_CONTACT_S::contactName

Indicates the display name of this contact.

5.4.2.4 size_t _MSG_CONTACT_S::extDataSize

Indicates the size of the extension data which `pExtData` point to. The unit is byte.

5.4.2.5 void* _MSG_CONTACT_S::pExtData

Extension data for contact.

Some data that is not covered in the structure can be put here, such as picture data and so on.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Contact.h](#)

5.5 _MSG_FILTER_ID_LIST_S Struct Reference

```
#include <Msg_Mapi_FilterRule.h>
```

5.5.1 Detailed Description

Represents a filter id list.

It is composed of various filter ids.

Data Fields

- [int iCount](#)
- [MSG_FILTER_ID_T * filterIds](#)

5.5.2 Field Documentation

5.5.2.1 int _MSG_FILTER_ID_LIST_S::iCount

Indicates the number of the filter id in the list.

5.5.2.2 MSG_FILTER_ID_T* _MSG_FILTER_ID_LIST_S::filterIds

Indicates the pointer to the filter id list.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_FilterRule.h](#)

5.6 `_MSG_FILTER_LIST_S` Struct Reference

```
#include <Msg_Mapi_FilterRule.h>
```

5.6.1 Detailed Description

Represents a filter list.

It is composed of various `MSG_FILTER_S`. The first filter in the list has the highest priority. Framework will check the filter list from the first filter to the last one. Once the message satisfies the filter, framework stops checking other filters in the list and returns the `autoStore` and `intercept` properties of the current filter.

Data Fields

- `int iCount`
- `MSG_FILTER_S * filters`

5.6.2 Field Documentation

5.6.2.1 `int _MSG_FILTER_LIST_S::iCount`

Indicates the number of the filters in the list.

5.6.2.2 `MSG_FILTER_S* _MSG_FILTER_LIST_S::filters`

The pointer to the filter list.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_FilterRule.h](#)

5.7 _MSG_FILTER_S Struct Reference

```
#include <Msg_Mapi_FilterRule.h>
```

5.7.1 Detailed Description

Represents a filter.

It represents the basic filter unit.

Data Fields

- [MSG_FILTER_ID_T filterId](#)
- [MSG_STRING_S filterName](#)
- [bool autoStore](#)
- [bool intercept](#)
- [int valueLength](#)
- [unsigned char * filterValue](#)

5.7.2 Field Documentation

5.7.2.1 MSG_FILTER_ID_T _MSG_FILTER_S::filterId

Indicates the filter type.

The filter Id indicates the filter type. There are some predefined filter ids. Please refer to [_MSG_FILTER_ID_E](#)

5.7.2.2 MSG_STRING_S _MSG_FILTER_S::filterName

Indicates the display name for the filter.

5.7.2.3 bool _MSG_FILTER_S::autoStore

Indicates whether the message should be automatically stored by framework, when it satisfies this filter.

If it is set `true`, the message will be automatically stored by framework when the message satisfies this filter. By default, this message will be saved to Inbox in the default account.

If it is set `false`, the message will not be stored by framework.

5.7.2.4 bool _MSG_FILTER_S::intercept

Indicates whether the message should be intercepted, when it satisfies this filter.

When the message satisfies this filter, framework will send the message to this filter associated session, then check the `intercept`:

If it is set `true`, the sessions whose priority are lower than this filter associated session will not receive this message.

If it is set `false`, framework will send the message to all the other sessions that are interested in this message.

For details about session priority, please refer to `MsgSetPriority` in *Messaging API* [2].

5.7.2.5 `int _MSG_FILTER_S::valueLength`

Indicates the length of the `filterValue`.

5.7.2.6 `unsigned char* _MSG_FILTER_S::filterValue`

The value of the filter.

For example, if the filter Id show that the message should be filtered by message type, the value maybe SMS, MMS.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_FilterRule.h](#)

5.8 _MSG_FOLDER_ID_LIST_S Struct Reference

```
#include <Msg_Mapi_Types.h>
```

5.8.1 Detailed Description

Represents a folder id list.

Data Fields

- [int iCount](#)
- [MSG_FOLDER_ID_T * folderIds](#)

5.8.2 Field Documentation

5.8.2.1 int _MSG_FOLDER_ID_LIST_S::iCount

The count of message ids

5.8.2.2 MSG_FOLDER_ID_T* _MSG_FOLDER_ID_LIST_S::folderIds

The pointer to the folder id list

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Types.h](#)

5.9 _MSG_FOLDER_S Struct Reference

```
#include <Msg_Mapi_Folder.h>
```

5.9.1 Detailed Description

Represents a folder.

For the more detailed information of folder, please refer to “Folder” in *Messaging API* [2] . This structure is extendable. Please refer to "Extendability" in *Messaging API* [2].

Data Fields

- [MSG_FOLDER_ID_T](#) folderId
- [MSG_FOLDER_ID_T](#) parentId
- [MSG_STRING_S](#) folderName
- [MSG_STORAGE_ID_T](#) storageId
- [MSG_SORT_TYPE_T](#) sortType
- [MSG_FOLDER_TYPE_T](#) folderType
- [size_t](#) commExtDataSize
- [void *](#) pCommExtData

5.9.2 Field Documentation

5.9.2.1 MSG_FOLDER_ID_T _MSG_FOLDER_S::folderId

Indicates folder id.

5.9.2.2 MSG_FOLDER_ID_T _MSG_FOLDER_S::parentId

Indicates parent folder id.

5.9.2.3 MSG_STRING_S _MSG_FOLDER_S::folderName

Indicates the folder name.

5.9.2.4 MSG_STORAGE_ID_T _MSG_FOLDER_S::storageId

Indicates the storage id in which this folder (metadata and extension data) is stored.

5.9.2.5 MSG_SORT_TYPE_T _MSG_FOLDER_S::sortType

Indicates the sort type for the children nodes under this folder.

5.9.2.6 MSG_FOLDER_TYPE_T _MSG_FOLDER_S::folderType

Indicates the folder type which can be Inbox/Outbox...

5.9.2.7 `size_t _MSG_FOLDER_S::commExtDataSize`

Indicates the size of `pCommExtData`

5.9.2.8 `void* _MSG_FOLDER_S::pCommExtData`

Used to add extension data for framework storage plugin.

For example, the extension data can be picture, max displaying size and so on.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Folder.h](#)

5.10 _MSG_FW_STORAGE_S Struct Reference

```
#include <Msg_Framework_Storage_Interface.h>
```

5.10.1 Detailed Description

Represents a framework storage plugin handle.

Data Fields

- [MsgFwStoAttachStorage pAttachStorage](#)
- [MsgFwStoDetachStorage pDetachStorage](#)
- [MsgFwStoCheckStatus pCheckStatus](#)
- [MsgFwStoInitStorage pInitStorage](#)
- [MsgFwStoGetStorageStatus pGetStorageStatus](#)
- [MsgFwStoAddMessage pAddMsg](#)
- [MsgFwStoUpdateMessage pUpdateMsg](#)
- [MsgFwStoDeleteMessage pDeleteMsg](#)
- [MsgFwStoDelAllMsgInFolder pDeleteAllMsg](#)
- [MsgFwStoCountMessage pCountMsg](#)
- [MsgFwStoCountUnreadMessage pCountUnreadMsg](#)
- [MsgFwStoQueryMessage pQueryMsg](#)
- [MsgFwStoGetMsgCommInfo pGetMsgCommonInfo](#)
- [MsgFwStoGetMessageParentId pGetMsgParentId](#)
- [MsgFwStoReleaseMsgCommon pReleaseMsgCommon](#)
- [MsgFwStoReleaseMessageList pReleaseMessageIdList](#)
- [MsgFwStoGetMessageIdsByStorage pGetMessageIdsByStorage](#)
- [MsgFwStoAddFolder pAddFolder](#)
- [MsgFwStoUpdateFolder pUpdateFolder](#)
- [MsgFwStoDeleteFolder pDeleteFolder](#)
- [MsgFwStoCountSubFolder pCountSubFolder](#)
- [MsgFwStoQueryFolder pQueryFolder](#)
- [MsgFwStoGetFolder pGetFolder](#)
- [MsgFwStoGetFolderParentId pGetFolderParentId](#)
- [MsgFwStoReleaseFolder pReleaseFolder](#)
- [MsgFwStoReleaseFolderIdList pReleaseFolderIdList](#)
- [MsgFwStoGetFolderByType pGetFolderByType](#)
- [MsgFwStoGetFolderIdsByStorage pGetFolderIdsByStorage](#)
- [MsgFwStoAddAccount pAddAccount](#)
- [MsgFwStoUpdateAccount pUpdateAccount](#)
- [MsgFwStoDeleteAccount pDeleteAccount](#)
- [MsgFwStoCountAccount pCountAccount](#)
- [MsgFwStoQueryAccount pQueryAccount](#)
- [MsgFwStoGetAccountIdsByRootId pGetAccountIdsByRootId](#)
- [MsgFwStoGetCommAccount pGetComAccount](#)
- [MsgFwStoReleaseCommAccount pReleaseComAccount](#)
- [MsgFwStoGetDefaultAccID pGetDefaultAccID](#)
- [MsgFwStoSetDefaultAccID pSetDefaultAccID](#)
- [MsgFwStoReleaseAccountIdList pReleaseAccountIdList](#)
- [MsgFwStoGetAccountIdsByStorage pGetAccountIdsByStorage](#)

5.10.2 Field Documentation

- 5.10.2.1 MsgFwStoAttachStorage _MSG_FW_STORAGE_S::pAttachStorage
- 5.10.2.2 MsgFwStoDetachStorage _MSG_FW_STORAGE_S::pDetachStorage
- 5.10.2.3 MsgFwStoCheckStatus _MSG_FW_STORAGE_S::pCheckStatus
- 5.10.2.4 MsgFwStoInitStorage _MSG_FW_STORAGE_S::pInitStorage
- 5.10.2.5 MsgFwStoGetStorageStatus _MSG_FW_STORAGE_S::pGetStorageStatus
- 5.10.2.6 MsgFwStoAddMessage _MSG_FW_STORAGE_S::pAddMsg
- 5.10.2.7 MsgFwStoUpdateMessage _MSG_FW_STORAGE_S::pUpdateMsg
- 5.10.2.8 MsgFwStoDeleteMessage _MSG_FW_STORAGE_S::pDeleteMsg
- 5.10.2.9 MsgFwStoDelAllMsgInFolder _MSG_FW_STORAGE_S::pDeleteAllMsg
- 5.10.2.10 MsgFwStoCountMessage _MSG_FW_STORAGE_S::pCountMsg
- 5.10.2.11 MsgFwStoCountUnreadMessage _MSG_FW_STORAGE_S::pCountUnreadMsg
- 5.10.2.12 MsgFwStoQueryMessage _MSG_FW_STORAGE_S::pQueryMsg
- 5.10.2.13 MsgFwStoGetMsgCommInfo _MSG_FW_STORAGE_S::pGetMsgCommonInfo
- 5.10.2.14 MsgFwStoGetMessageParentId _MSG_FW_STORAGE_S::pGetMsgParentId
- 5.10.2.15 MsgFwStoReleaseMsgCommon _MSG_FW_STORAGE_S::pReleaseMsgCommon
- 5.10.2.16 MsgFwStoReleaseMessageList _MSG_FW_STORAGE_S::pReleaseMessageIdList
- 5.10.2.17 MsgFwStoGetMessageIdsByStorage _MSG_FW_STORAGE_S::pGetMessageIdsByStorage
- 5.10.2.18 MsgFwStoAddFolder _MSG_FW_STORAGE_S::pAddFolder
- 5.10.2.19 MsgFwStoUpdateFolder _MSG_FW_STORAGE_S::pUpdateFolder
- 5.10.2.20 MsgFwStoDeleteFolder _MSG_FW_STORAGE_S::pDeleteFolder
- 5.10.2.21 MsgFwStoCountSubFolder _MSG_FW_STORAGE_S::pCountSubFolder
- 5.10.2.22 MsgFwStoQueryFolder _MSG_FW_STORAGE_S::pQueryFolder
- 5.10.2.23 MsgFwStoGetFolder _MSG_FW_STORAGE_S::pGetFolder
- 5.10.2.24 MsgFwStoGetFolderParentId _MSG_FW_STORAGE_S::pGetFolderParentId
- 5.10.2.25 MsgFwStoReleaseFolder _MSG_FW_STORAGE_S::pReleaseFolder
- 5.10.2.26 MsgFwStoReleaseFolderIdList _MSG_FW_STORAGE_S::pReleaseFolderIdList
- 5.10.2.27 MsgFwStoGetFolderByType _MSG_FW_STORAGE_S::pGetFolderByType
- 5.10.2.28 MsgFwStoGetFolderIdsByStorage _MSG_FW_STORAGE_S::pGetFolderIdsByStorage
- 5.10.2.29 MsgFwStoAddAccount _MSG_FW_STORAGE_S::pAddAccount

- [Msg_Framework_Storage_Interface.h](#)

5.11 `_MSG_MESSAGE_ID_LIST_S` Struct Reference

```
#include <Msg_Mapi_Types.h>
```

5.11.1 Detailed Description

Represents a message id list.

Data Fields

- `int iCount`
- `MSG_MESSAGE_ID_T * msgIds`

5.11.2 Field Documentation

5.11.2.1 `int _MSG_MESSAGE_ID_LIST_S::iCount`

The count of message ids

5.11.2.2 `MSG_MESSAGE_ID_T* _MSG_MESSAGE_ID_LIST_S::msgIds`

The pointer to the message id list

The documentation for this struct was generated from the following file:

- `Msg_Mapi_Types.h`

5.12 _MSG_MESSAGE_REQUEST_S Struct Reference

```
#include <Msg_Mapi_Message_Request.h>
```

5.12.1 Detailed Description

Represents a request that will be sent to messaging framework.

Applications compose a request with this structure and send it to message framework via a connected session. This request Id is used to manage the request by framework.

This structure is extendable. Please refer to “Extendability” in *Messaging API* [2].

Data Fields

- [MSG_REQUEST_TYPE_T reqType](#)
- [MSG_MESSAGE_TYPE_S msgType](#)
- [MSG_REQUEST_ID_T reqId](#)
- [MSG_MESSAGE_S * pMsg](#)
- [size_t extDataSize](#)
- [void * pExtData](#)

5.12.2 Field Documentation

5.12.2.1 MSG_REQUEST_TYPE_T _MSG_MESSAGE_REQUEST_S::reqType

Indicates the type of request.

Some predefined types: [MSG_REQUEST_TYPE_SEND](#), [MSG_REQUEST_TYPE_SENDSAVE](#)

5.12.2.2 MSG_MESSAGE_TYPE_S _MSG_MESSAGE_REQUEST_S::msgType

Indicates the message type.

Indicates the type of the message such as SMS, MMS, Email etc.

5.12.2.3 MSG_REQUEST_ID_T _MSG_MESSAGE_REQUEST_S::reqId

Indicates the request Id, the unique Id of the request.

When applications use `MsgSubmitReq` (see *Messaging API* [2]) to submit a request to framework, this value will be set by framework and returned to applications. Applications should keep this id to trace the request's status or cancel the request later.

5.12.2.4 MSG_MESSAGE_S* _MSG_MESSAGE_REQUEST_S::pMsg

The pointer to message structure.

When applications submit sending message request, this pointer shall point to the message that should be sent.

5.12.2.5 `size_t _MSG_MESSAGE_REQUEST_S::extDataSize`

Indicates the size of the extension data that `pExtData` point to. The unit is byte.

5.12.2.6 `void* _MSG_MESSAGE_REQUEST_S::pExtData`

Used to add extension data for transport plugin.

For example, this field can be used to extend new request for specific service.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Message_Request.h](#)

5.13 _MSG_MESSAGE_S Struct Reference

```
#include <Msg_Mapi_Message.h>
```

5.13.1 Detailed Description

Represents a message.

This structure includes the common information of various specific service message, such as SMS, MMS, Email and i-mode message.

This structure is extendable. Please refer to “Extendability” in *Messaging API* [2].

Data Fields

- [MSG_MESSAGE_ID_T](#) msgId
- [MSG_FOLDER_ID_T](#) parentId
- [MSG_MESSAGE_TYPE_S](#) msgType
- [MSG_ACCOUNT_ID_T](#) accountId
- [MSG_STORAGE_ID_T](#) storageId
- [MSG_STRING_S](#) subject
- [MSG_CONTACT_S](#) from
- [MSG_CONTACT_LIST_S](#) to
- [MSG_STRING_S](#) displaySubject
- [MSG_STRING_S](#) displayFrom
- [MSG_STRING_S](#) displayTo
- [time_t](#) displayTime
- [size_t](#) size
- [MSG_STATUS_T](#) status
- [MSG_STATUS_T](#) deliveryStatus
- [bool](#) hasAttachment
- [bool](#) hasDrmContent
- [MSG_MESSAGE_PRIORITY_T](#) priority
- [size_t](#) commExtDataSize
- [void *](#) pCommExtData
- [size_t](#) extDataSize
- [void *](#) pExtData

5.13.2 Field Documentation

5.13.2.1 MSG_MESSAGE_ID_T _MSG_MESSAGE_S::msgId

Indicates the message Id of this message.

5.13.2.2 MSG_FOLDER_ID_T _MSG_MESSAGE_S::parentId

Indicates the parent folder ID.

The parent is a folder.

5.13.2.3 MSG_MESSAGE_TYPE_S _MSG_MESSAGE_S::msgType

Indicates the message type.

Indicates the type of the message such as SMS, MMS, Email, i-mode etc.

5.13.2.4 MSG_ACCOUNT_ID_T _MSG_MESSAGE_S::accountId

Indicates which account the message associated.

5.13.2.5 MSG_STORAGE_ID_T _MSG_MESSAGE_S::storageId

Indicates where this message (metadata and extension data) is stored.

If it is set [MSG_STORAGE_DEFAULT](#), the message will be saved according to defaultMsgStorageId in its associated account. Otherwise, the message will be saved according this value.

5.13.2.6 MSG_STRING_S _MSG_MESSAGE_S::subject

Indicates the subject of the message.

5.13.2.7 MSG_CONTACT_S _MSG_MESSAGE_S::from

Indicates where the message comes from.

5.13.2.8 MSG_CONTACT_LIST_S _MSG_MESSAGE_S::to

Indicates where the message will send to.

5.13.2.9 MSG_STRING_S _MSG_MESSAGE_S::displaySubject

Indicates the display content for subject.

5.13.2.10 MSG_STRING_S _MSG_MESSAGE_S::displayFrom

Indicates the display content for from.

5.13.2.11 MSG_STRING_S _MSG_MESSAGE_S::displayTo

Indicates the display content for to.

5.13.2.12 time_t _MSG_MESSAGE_S::displayTime

Indicates the display content for time.

For example, if the message is an incoming message, this value should be the time when the message arrived. Framework is responsible to set this value in this case.

If the message is an outgoing message, this value should be the time when the message be sent out. Application is responsible to set this value in this case.

5.13.2.13 `size_t _MSG_MESSAGE_S::size`

Indicates the size of the message. The unit is byte.

5.13.2.14 `MSG_STATUS_T _MSG_MESSAGE_S::status`

Indicates the message's local status, such as read/unread/replied/forwarded/protected ...

5.13.2.15 `MSG_STATUS_T _MSG_MESSAGE_S::deliveryStatus`

Indicates the delivery status of the message.

5.13.2.16 `bool _MSG_MESSAGE_S::hasAttachment`

Indicates whether the message has attachment.

If the message has attachment, it should be set `true`.

If the message does not have attachment, it should be set `false`.

5.13.2.17 `bool _MSG_MESSAGE_S::hasDrmContent`

Indicates whether the message attaches the DRM contents.

If it is set `true`, the message attaches the DRM contents.

If it is set `false`, the message does not attaches the DRM contents.

5.13.2.18 `MSG_MESSAGE_PRIORITY_T _MSG_MESSAGE_S::priority`

Indicates the priority of this message.

5.13.2.19 `size_t _MSG_MESSAGE_S::commExtDataSize`

Indicates the size of the data that `pCommExtData` point to. The unit is byte.

5.13.2.20 `void* _MSG_MESSAGE_S::pCommExtData`

Used to add extension data for framework storage plugin.

This field is the extension data for framework storage plugin. For example, whether the message is replied can be regarded as an common extension attribute.

5.13.2.21 `size_t _MSG_MESSAGE_S::extDataSize`

Indicates the size of the data that `pExtData` point to. The unit is byte.

5.13.2.22 void* _MSG_MESSAGE_S::pExtData

Used to add extension data for message storage/transport plugin.

This field is the extension data for storage/transport plugin. It is the place to put such message payload information as body and so on.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Message.h](#)

5.14 _MSG_MESSAGE_STATUS_S Struct Reference

```
#include <Msg_Mapi_Message_Status.h>
```

5.14.1 Detailed Description

Represents a message status.

This structure contains the information about the status of the request that application has submitted to framework.

This structure is extendable. Please refer to “Extendability” in *Messaging API* [2].

Data Fields

- [MSG_REQUEST_ID_T reqId](#)
- [MSG_ERROR_T error](#)
- [MSG_STATUS_T status](#)
- [bool isFinal](#)
- [MSG_MESSAGE_ID_T msgId](#)
- [size_t extDataSize](#)
- [void * pExtData](#)

5.14.2 Field Documentation

5.14.2.1 MSG_REQUEST_ID_T _MSG_MESSAGE_STATUS_S::reqId

Indicates the corresponding request Id.

5.14.2.2 MSG_ERROR_T _MSG_MESSAGE_STATUS_S::error

Indicates the error code.

5.14.2.3 MSG_STATUS_T _MSG_MESSAGE_STATUS_S::status

Indicates the status of the corresponding request.

5.14.2.4 bool _MSG_MESSAGE_STATUS_S::isFinal

Indicates the request is finished or not. If the request is finished, no more status of this request will be sent to client and the request information will be removed from messaging framework.

5.14.2.5 MSG_MESSAGE_ID_T _MSG_MESSAGE_STATUS_S::msgId

Indicates the saved message id, it's applicable when status is MSG_STATUS_SAVED_NOTSENT or MSG_STATUS_SENT_SAVED.

5.14.2.6 `size_t _MSG_MESSAGE_STATUS_S::extDataSize`

Indicates the size of the extension data that `pExtData` point to. The unit is byte.

5.14.2.7 `void* _MSG_MESSAGE_STATUS_S::pExtData`

Used to add extension data for transport plugin.

For example, this field can be used to extend new status for specific service.

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Message_Status.h](#)

5.15 _MSG_MESSAGE_TYPE_LIST_S Struct Reference

```
#include <Msg_Mapi_Types.h>
```

5.15.1 Detailed Description

Represents a message types list.

Data Fields

- [int iCount](#)
- [MSG_MESSAGE_TYPE_S * pMsgTypes](#)

5.15.2 Field Documentation

5.15.2.1 int _MSG_MESSAGE_TYPE_LIST_S::iCount

The count of message types

5.15.2.2 MSG_MESSAGE_TYPE_S* _MSG_MESSAGE_TYPE_LIST_S::pMsgTypes

The pointer to message types list

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Types.h](#)

5.16 _MSG_MESSAGE_TYPE_S Struct Reference

```
#include <Msg_Mapi_Types.h>
```

5.16.1 Detailed Description

Represents a message type.

This structure includes main type and sub type.

Data Fields

- [MSG_MAIN_TYPE_T](#) mainType
- [MSG_SUB_TYPE_T](#) subType

5.16.2 Field Documentation

5.16.2.1 MSG_MAIN_TYPE_T _MSG_MESSAGE_TYPE_S::mainType

Message main type

5.16.2.2 MSG_SUB_TYPE_T _MSG_MESSAGE_TYPE_S::subType

Message sub type

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Types.h](#)

5.17 _MSG_PLG_STORAGE_S Struct Reference

```
#include <Msg_Plugin_Storage_Interface.h>
```

5.17.1 Detailed Description

Represents a message storage plugin handle.

Data Fields

- [MsgPlgStoSetFwStoHandle pSetFwStoHandle](#)
- [MsgPlgStoAttachStorage pAttachStorage](#)
- [MsgPlgStoDetachStorage pDetachStorage](#)
- [MsgPlgStoCheckStatus pCheckStatus](#)
- [MsgPlgStoInitStorage pInitStorage](#)
- [MsgPlgStoGetStorageStatus pGetStorageStatus](#)
- [MsgPlgStoAddMessage pAddMsg](#)
- [MsgPlgStoUpdateMessage pUpdateMsg](#)
- [MsgPlgStoDeleteMessage pDeleteMsg](#)
- [MsgPlgStoDelAllMsgInFolder pDelAllMsg](#)
- [MsgPlgStoGetMessage pGetMsg](#)
- [MsgPlgStoReleaseMessage pReleaseMsg](#)
- [MsgPlgStoAddAccount pAddAccount](#)
- [MsgPlgStoUpdateAccount pUpdateAccount](#)
- [MsgPlgStoDeleteAccount pDeleteAccount](#)
- [MsgPlgStoGetAccount pGetAccount](#)
- [MsgPlgStoReleaseAccount pReleaseAccount](#)

5.17.2 Field Documentation

- 5.17.2.1 `MsgPlgStoSetFwStoHandle _MSG_PLG_STORAGE_S::pSetFwStoHandle`
- 5.17.2.2 `MsgPlgStoAttachStorage _MSG_PLG_STORAGE_S::pAttachStorage`
- 5.17.2.3 `MsgPlgStoDetachStorage _MSG_PLG_STORAGE_S::pDetachStorage`
- 5.17.2.4 `MsgPlgStoCheckStatus _MSG_PLG_STORAGE_S::pCheckStatus`
- 5.17.2.5 `MsgPlgStoInitStorage _MSG_PLG_STORAGE_S::pInitStorage`
- 5.17.2.6 `MsgPlgStoGetStorageStatus _MSG_PLG_STORAGE_S::pGetStorageStatus`
- 5.17.2.7 `MsgPlgStoAddMessage _MSG_PLG_STORAGE_S::pAddMsg`
- 5.17.2.8 `MsgPlgStoUpdateMessage _MSG_PLG_STORAGE_S::pUpdateMsg`
- 5.17.2.9 `MsgPlgStoDeleteMessage _MSG_PLG_STORAGE_S::pDeleteMsg`
- 5.17.2.10 `MsgPlgStoDelAllMsgInFolder _MSG_PLG_STORAGE_S::pDelAllMsg`
- 5.17.2.11 `MsgPlgStoGetMessage _MSG_PLG_STORAGE_S::pGetMsg`
- 5.17.2.12 `MsgPlgStoReleaseMessage _MSG_PLG_STORAGE_S::pReleaseMsg`
- 5.17.2.13 `MsgPlgStoAddAccount _MSG_PLG_STORAGE_S::pAddAccount`
- 5.17.2.14 `MsgPlgStoUpdateAccount _MSG_PLG_STORAGE_S::pUpdateAccount`
- 5.17.2.15 `MsgPlgStoDeleteAccount _MSG_PLG_STORAGE_S::pDeleteAccount`
- 5.17.2.16 `MsgPlgStoGetAccount _MSG_PLG_STORAGE_S::pGetAccount`
- 5.17.2.17 `MsgPlgStoReleaseAccount _MSG_PLG_STORAGE_S::pReleaseAccount`

The documentation for this struct was generated from the following file:

- [Msg_Plugin_Storage_Interface.h](#)

5.18 _MSG_PLUGIN_FILTER_MANAGER_S Struct Reference

```
#include <Msg_Plugin_Filter_Interface.h>
```

5.18.1 Detailed Description

Represents a filter manager.

This structure contains the function pointers that filter rule plugin MUST implement. Filter rule plugin MUST implement all the functions and assemble them into this structure, then provide it to user.

Data Fields

- [MsgPlgFilterInitialize pfIntialize](#)
- [MsgPlgFilterFinalize pfFinalize](#)
- [MsgPlgFilterCheckRules pfCheckRules](#)

5.18.2 Field Documentation

5.18.2.1 MsgPlgFilterInitialize _MSG_PLUGIN_FILTER_MANAGER_S::pfIntialize

The function pointer of initialize function.

5.18.2.2 MsgPlgFilterFinalize _MSG_PLUGIN_FILTER_MANAGER_S::pfFinalize

The function pointer of finalize function.

5.18.2.3 MsgPlgFilterCheckRules _MSG_PLUGIN_FILTER_MANAGER_S::pfCheckRules

The function pointer of checkRules function.

The documentation for this struct was generated from the following file:

- [Msg_Plugin_Filter_Interface.h](#)

5.19 _MSG_PLUGIN_TRANSPORT_S Struct Reference

```
#include <Msg_Plugin_Transport_Interface.h>
```

5.19.1 Detailed Description

Represents a transport plugin handle.

This structure contains the function pointers that transport plugin MUST implment.

Data Fields

- [MsgPlgTransInitialize pfInitialize](#)
- [MsgPlgTransFinalize pfFinalize](#)
- [MsgPlgTransSubmitRequest pfSubmitRequest](#)
- [MsgPlgTransCancelRequest pfCancelRequest](#)
- [MsgPlgTransDiscardMsg pfDiscardMsg](#)
- [MsgPlgTransRegisterListener pfRegisterListener](#)

5.19.2 Field Documentation

5.19.2.1 MsgPlgTransInitialize _MSG_PLUGIN_TRANSPORT_S::pfInitialize

The function pointer of initialize.

5.19.2.2 MsgPlgTransFinalize _MSG_PLUGIN_TRANSPORT_S::pfFinalize

The function pointer of finalize.

5.19.2.3 MsgPlgTransSubmitRequest _MSG_PLUGIN_TRANSPORT_S::pfSubmitRequest

The function pointer of submit request.

5.19.2.4 MsgPlgTransCancelRequest _MSG_PLUGIN_TRANSPORT_S::pfCancelRequest

The function pointer of cancel request.

5.19.2.5 MsgPlgTransDiscardMsg _MSG_PLUGIN_TRANSPORT_S::pfDiscardMsg

The function pointer of discard message.

5.19.2.6 MsgPlgTransRegisterListener _MSG_PLUGIN_TRANSPORT_S::pfRegisterListener

The function pointer of register listener.

The documentation for this struct was generated from the following file:

- [Msg_Plugin_Transport_Interface.h](#)

5.20 _MSG_QUERY_ID_LIST_S Struct Reference

```
#include <Msg_Mapi_QueryRule.h>
```

5.20.1 Detailed Description

Represents a query Id list.

Data Fields

- [int iCount](#)
- [MSG_QUERY_ID_T * queryIds](#)

5.20.2 Field Documentation

5.20.2.1 [int _MSG_QUERY_ID_LIST_S::iCount](#)

Indicates the count of the query Id in the list

5.20.2.2 [MSG_QUERY_ID_T* _MSG_QUERY_ID_LIST_S::queryIds](#)

Indicates the pointer to the query Id list

The documentation for this struct was generated from the following file:

- [Msg_Mapi_QueryRule.h](#)

5.21 `_MSG_QUERY_LIST_S` Struct Reference

```
#include <Msg_Mapi_QueryRule.h>
```

5.21.1 Detailed Description

Represents a query list.

Data Fields

- `int iCount`
- `MSG_QUERY_S * queries`

5.21.2 Field Documentation

5.21.2.1 `int _MSG_QUERY_LIST_S::iCount`

Indicates the count of the query rules in the list

5.21.2.2 `MSG_QUERY_S* _MSG_QUERY_LIST_S::queries`

Indicates the pointer to the query list

The documentation for this struct was generated from the following file:

- `Msg_Mapi_QueryRule.h`

5.22 _MSG_QUERY_S Struct Reference

```
#include <Msg_Mapi_QueryRule.h>
```

5.22.1 Detailed Description

Represents a query rule.

When applications need to query all SMS with subject containing some values, a [MSG_QUERY_ID_T](#) can be added as "MSG_QUERY_SMS_SUBJECT_CONTAIN".

Refer to the following code:

```
MSG_QUERY_S queryRule1;  
queryRule.queryId = MSG_QUERY_BY_SUBJECT;  
queryRule.queryValue = "SearchingValue";  
queryRule.valueLength = strlen(queryRule.queryValue);
```

Data Fields

- [MSG_QUERY_ID_T](#) queryId
- int valueLength
- unsigned char * queryValue

5.22.2 Field Documentation

5.22.2.1 MSG_QUERY_ID_T _MSG_QUERY_S::queryId

Indicates what kind of rules should be taken when framework filters the messages

5.22.2.2 int _MSG_QUERY_S::valueLength

Indicates the length of the queryValue

5.22.2.3 unsigned char* _MSG_QUERY_S::queryValue

Indicates the value of the query

The documentation for this struct was generated from the following file:

- [Msg_Mapi_QueryRule.h](#)

5.23 _MSG_SORT_RULE_S Struct Reference

```
#include <Msg_Mapi_QueryRule.h>
```

5.23.1 Detailed Description

Represents a sort rule.

The sort rule structure includes sort type and sort order (ascending or descending). Applications can use the sort rule when querying messages.

Data Fields

- [MSG_SORT_TYPE_T](#) sortType
- bool [isAscending](#)

5.23.2 Field Documentation

5.23.2.1 MSG_SORT_TYPE_T _MSG_SORT_RULE_S::sortType

Indicates the sort type

5.23.2.2 bool _MSG_SORT_RULE_S::isAscending

Indicates the sort order: ascending or descending

The documentation for this struct was generated from the following file:

- [Msg_Mapi_QueryRule.h](#)

5.24 _MSG_SORT_TYPE_LIST_S Struct Reference

```
#include <Msg_Mapi_QueryRule.h>
```

5.24.1 Detailed Description

Represents a sort type list.

Data Fields

- [int iCount](#)
- [MSG_SORT_TYPE_T * sortTypes](#)

5.24.2 Field Documentation

5.24.2.1 [int _MSG_SORT_TYPE_LIST_S::iCount](#)

Indicates the count of the sort type in the list

5.24.2.2 [MSG_SORT_TYPE_T* _MSG_SORT_TYPE_LIST_S::sortTypes](#)

Indicates the pointer to the sort type list

The documentation for this struct was generated from the following file:

- [Msg_Mapi_QueryRule.h](#)

5.25 `_MSG_STORAGE_ID_LIST_S` Struct Reference

```
#include <Msg_Mapi_Types.h>
```

5.25.1 Detailed Description

Represents a storage id list.

Data Fields

- `int iCount`
- `MSG_STORAGE_ID_T * pStorageIds`

5.25.2 Field Documentation

5.25.2.1 `int _MSG_STORAGE_ID_LIST_S::iCount`

The count of storage ids

5.25.2.2 `MSG_STORAGE_ID_T* _MSG_STORAGE_ID_LIST_S::pStorageIds`

The pointer to storage id list

The documentation for this struct was generated from the following file:

- `Msg_Mapi_Types.h`

5.26 _MSG_STRING_S Struct Reference

```
#include <Msg_Mapi_Types.h>
```

5.26.1 Detailed Description

Represents a string.

Data Fields

- [MSG_ENCODE_TYPE_T](#) `encodeType`
- [int](#) `length`
- `unsigned char *` [pString](#)

5.26.2 Field Documentation

5.26.2.1 `MSG_ENCODE_TYPE_T _MSG_STRING_S::encodeType`

The encoding type of the string

5.26.2.2 `int _MSG_STRING_S::length`

The length of string value

5.26.2.3 `unsigned char* _MSG_STRING_S::pString`

The string value

The documentation for this struct was generated from the following file:

- [Msg_Mapi_Types.h](#)

5.27 `_MSG_TRANSPORT_LISTENER_S` Struct Reference

```
#include <Msg_Plugin_Transport_Interface.h>
```

5.27.1 Detailed Description

Represents a transport listener handle.

This structure contains the function pointers that transport listener of framework MUST implement.

Data Fields

- [MsgPlgTransOnStatusChanged pfStatusChangedCallback](#)
- [MsgPlgTransOnMsgIncoming pfReceiveMsgCallback](#)

5.27.2 Field Documentation

5.27.2.1 `MsgPlgTransOnStatusChanged _MSG_TRANSPORT_LISTENER_S::pfStatusChangedCallback`

The function pointer of status changed callback.

5.27.2.2 `MsgPlgTransOnMsgIncoming _MSG_TRANSPORT_LISTENER_S::pfReceiveMsgCallback`

The function pointer of receive message callback.

The documentation for this struct was generated from the following file:

- [Msg_Plugin_Transport_Interface.h](#)

Chapter 6

File Documentation

6.1 ComponentOverview.h File Reference

6.1.1 Detailed Description

Component Programming Interface Overview Header Template for Doxygen.

This file contains configuration information for documentation generation; it does not contain any API-specific information.

Defines

- #define [COMPONENT_OVERVIEW_H](#)

6.1.2 Define Documentation

6.1.2.1 #define COMPONENT_OVERVIEW_H

6.2 Msg_Framework_Storage_Interface.h File Reference

6.2.1 Detailed Description

Defines the interfaces that framework storage plugin MUST implement.

These functions can be divided into four categories:

- Storage management.
- Message storage.
- Folder storage.
- Account storage.

```
#include "Msg_Mapi_Types.h"
#include "Msg_Mapi_QueryRule.h"
#include "Msg_Mapi_Storage_Manager.h"
#include "Msg_Plugin_Types.h"
#include "Msg_Mapi_Message.h"
#include "Msg_Mapi_Folder.h"
#include "Msg_Mapi_Account.h"
```

Data Structures

- struct [_MSG_FW_STORAGE_S](#)
Represents a framework storage plugin handle.

Typedefs

- typedef [MSG_ERROR_T](#)(* [MsgFwStoAttachStorage](#))(MSG_STORAGE_ID_T StorageId)
The prototype for attaching a specific storage by storage id.
- typedef [MSG_ERROR_T](#)(* [MsgFwStoDetachStorage](#))(MSG_STORAGE_ID_T StorageId)
The prototype for detaching a specific storage by storage id.
- typedef [MSG_ERROR_T](#)(* [MsgFwStoCheckStatus](#))(MSG_STORAGE_ID_T StorageId, [MSG_PLG_STORAGE_STATUS_T](#) *pStatus)
The prototype for checking the status of the storage by storage id.
- typedef [MSG_ERROR_T](#)(* [MsgFwStoInitStorage](#))(MSG_STORAGE_ID_T StorageId)
The prototype for initializing the storage.
- typedef [MSG_ERROR_T](#)(* [MsgFwStoGetStorageStatus](#))(MSG_STORAGE_ID_T storageId, unsigned long long *totalSize, unsigned long long *usedSize, [MSG_SIZE_UNIT_T](#) *unit)
Gets framework storage status by storage id.

- typedef `MSG_ERROR_T(* MsgFwStoAddMessage )(MSG_MESSAGE_S *pMsg)`
The prototype for adding a message into storage.
- typedef `MSG_ERROR_T(* MsgFwStoUpdateMessage )(const MSG_MESSAGE_S *pMsg)`
The prototype for updating a message in storage.
- typedef `MSG_ERROR_T(* MsgFwStoDeleteMessage )(MSG_MESSAGE_ID_T MsgId)`
The prototype for deleting a message.
- typedef `MSG_ERROR_T(* MsgFwStoDelAllMsgInFolder )(MSG_FOLDER_ID_T FolderId)`
Deletes all messages in the specified folder.
- typedef `MSG_ERROR_T(* MsgFwStoCountMessage )(MSG_FOLDER_ID_T FolderId, bool bRecursive, const MSG_MESSAGE_TYPE_S *pMsgType, int *pCount)`
The prototype for getting the number of messages under the specified folder.
- typedef `MSG_ERROR_T(* MsgFwStoCountUnreadMessage )(MSG_FOLDER_ID_T FolderId, bool bRecursive, const MSG_MESSAGE_TYPE_S *pMsgType, int *pCount)`
The prototype for getting the number of unread messages under the specified folder.
- typedef `MSG_ERROR_T(* MsgFwStoQueryMessage )(MSG_FOLDER_ID_T FolderId, bool bRecursive, int iBeginIndex, int iCount, const MSG_SORT_RULE_S *pSortRule, const MSG_QUERY_LIST_S *pQueryList, MSG_MESSAGE_ID_LIST_S *pMsgIdList)`
The prototype for getting the matched message ids under the specified folder.
- typedef `MSG_ERROR_T(* MsgFwStoGetMsgCommInfo )(MSG_MESSAGE_ID_T MsgId, MSG_MESSAGE_S **ppMsg)`
The prototype for returning the message common information.
- typedef `MSG_ERROR_T(* MsgFwStoGetMessageParentId )(MSG_MESSAGE_ID_T MsgId, MSG_FOLDER_ID_T *pParentFolderId)`
The prototype for returning the folder id in which the message exists.
- typedef `void(* MsgFwStoReleaseMsgCommon )(MSG_MESSAGE_S *pMsg)`
The prototype for releasing the memory of MSG_MESSAGE_S allocated in [MsgFwStoGetMsgCommInfo](#).
- typedef `void(* MsgFwStoReleaseMessageList )(MSG_MESSAGE_ID_LIST_S *pMsgIdList)`
The prototype for releasing the memory of the pointer in MSG_MESSAGE_ID_LIST_S.
- typedef `MSG_ERROR_T(* MsgFwStoGetMessageIdsByStorage )(MSG_STORAGE_ID_T StorageId, MSG_MESSAGE_ID_LIST_S *pMsgIds)`
Returns the message Ids by storage Id.
- typedef `MSG_ERROR_T(* MsgFwStoAddFolder )(MSG_FOLDER_S *pFolder)`
The prototype for adding a new folder.
- typedef `MSG_ERROR_T(* MsgFwStoUpdateFolder )(const MSG_FOLDER_S *pFolder)`
The prototype for updating a folder.
- typedef `MSG_ERROR_T(* MsgFwStoDeleteFolder )(MSG_FOLDER_ID_T folderId, bool recursive)`

The prototype for deleting a folder.

- typedef `MSG_ERROR_T(* MsgFwStoCountSubFolder )(MSG_FOLDER_ID_T folderId, bool recursive, int *pCount)`

The prototype for getting the number of sub folders under the specified folder.

- typedef `MSG_ERROR_T(* MsgFwStoQueryFolder )(MSG_FOLDER_ID_T folderId, int iBeginPos, int iCount, bool recursive, MSG_FOLDER_ID_LIST_S *pFolderIdList)`

The prototype for querying the sub folder ids under the specified folder.

- typedef `MSG_ERROR_T(* MsgFwStoGetFolder )(MSG_FOLDER_ID_T folderId, MSG_FOLDER_S **ppFolderInfo)`

The prototype for getting the folder information of the specified folder.

- typedef `MSG_ERROR_T(* MsgFwStoGetFolderParentId )(MSG_FOLDER_ID_T folderId, MSG_FOLDER_ID_T *pParentFolderId)`

The prototype for getting the parent folder id of a folder.

- typedef `void(* MsgFwStoReleaseFolder )(MSG_FOLDER_S *pFolderInfo)`

The prototype for releasing the memory block of a folder.

- typedef `void(* MsgFwStoReleaseFolderIdList )(MSG_FOLDER_ID_LIST_S *pFolderIdList)`

The prototype for releasing a memory block of folder id list.

- typedef `MSG_ERROR_T(* MsgFwStoGetFolderByType )(MSG_FOLDER_ID_T rootFolderId, MSG_FOLDER_TYPE_T folderType, MSG_FOLDER_ID_LIST_S *pFolderIdList)`

The prototype for getting the folder by folder type for the specified root folder.

- typedef `MSG_ERROR_T(* MsgFwStoGetFolderIdsByStorage )(MSG_STORAGE_ID_T storageId, MSG_FOLDER_ID_LIST_S *pFolderIdList)`

The prototype for getting all the folder id list on a specified storage.

- typedef `MSG_ERROR_T(* MsgFwStoAddAccount )(MSG_ACCOUNT_S *pAccount)`

The prototype for adding account common information into framework storage plugin.

- typedef `MSG_ERROR_T(* MsgFwStoUpdateAccount )(const MSG_ACCOUNT_S *pAccount)`

The prototype for updating account common information.

- typedef `MSG_ERROR_T(* MsgFwStoDeleteAccount )(MSG_ACCOUNT_ID_T accountId)`

The prototype for deleting account common information.

- typedef `MSG_ERROR_T(* MsgFwStoCountAccount )(const MSG_MESSAGE_TYPE_S *pMsgType, int *pCount)`

The prototype for getting the accounts number of the specified message type.

- typedef `MSG_ERROR_T(* MsgFwStoQueryAccount )(const MSG_MESSAGE_TYPE_S *pMsgType, MSG_ACCOUNT_ID_LIST_S *pAccountIdList)`

The prototype for querying the account id list of the specified message type.

- typedef `MSG_ERROR_T(* MsgFwStoGetAccountIdsByRootId )(MSG_FOLDER_ID_T rootFolderId, MSG_ACCOUNT_ID_LIST_S *pAccountIdList)`

The prototype for getting account ids by root folder id.

- typedef MSG_ERROR_T(* MsgFwStoGetCommAccount)(MSG_ACCOUNT_ID_T accountId, MSG_ACCOUNT_S **ppAccount)

The prototype for getting account common information.

- typedef void(* MsgFwStoReleaseCommAccount)(MSG_ACCOUNT_S *pAccount)

The prototype for releasing account common information.

- typedef MSG_ERROR_T(* MsgFwStoGetDefaultAccID)(const MSG_MESSAGE_TYPE_S *pMsgType, MSG_ACCOUNT_ID_T *pAccountId)

The prototype for getting the default account id.

- typedef MSG_ERROR_T(* MsgFwStoSetDefaultAccID)(const MSG_MESSAGE_TYPE_S *pMsgType, MSG_ACCOUNT_ID_T accountId)

The prototype for setting the default account id.

- typedef void(* MsgFwStoReleaseAccountIdList)(MSG_ACCOUNT_ID_LIST_S *pAccountIdList)

The prototype for release the memory of account id list.

- typedef MSG_ERROR_T(* MsgFwStoGetAccountIdsByStorage)(MSG_STORAGE_ID_T storageId, MSG_ACCOUNT_ID_LIST_S *pAccountIdList)

The prototype for getting all the account ids on a specified storage.

- typedef struct _MSG_FW_STORAGE_S MSG_FW_STORAGE_S

Functions

- MSG_ERROR_T MsgFwStoCreateHandle (MSG_FW_STORAGE_S *hFwStoHandle)

Creates a framework storage plugin handle.

- MSG_ERROR_T MsgFwStoDestroyHandle ()

Destroys a framework storage plugin handle.

6.2.2 Typedef Documentation

6.2.2.1 typedef struct _MSG_FW_STORAGE_S MSG_FW_STORAGE_S

6.2.2.2 typedef MSG_ERROR_T(* MsgFwStoAddAccount)(MSG_ACCOUNT_S *pAccount)

The prototype for adding account common information into framework storage plugin.

This function is to add account common information into framework storage plugin. Mandatory fields of account structure MUST be valid, otherwise the function MUST be failed. The mandatory fields for adding operation are accountId, msgType, rootFolderId, storageId, defaultMsgStorageId.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#).

Parameters:

← *pAccount* is a pointer to the data user defined for this new account.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER pAccount is NULL.

MSG_ERR_INVALID_ACCOUNT Some mandatory fields of account structure are not filled or not valid.

MSG_ERR_STORAGE_FULL Storage is FULL.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.3 typedef MSG_ERROR_T(* MsgFwStoAddFolder)(MSG_FOLDER_S *pFolder)

The prototype for adding a new folder.

This function is to create a folder under the specified folder in framework storage plugin. Mandatory fields of folder structure MUST be valid, otherwise the function MUST be failed. The mandatory fields for creating operation are `folderId`, `parentId`, `storageId`. If the `sortType` is invalid, the default value MSG_SORT_BY_UNKNOWN MUST be set.

Parameters:

← *pFolder* is a pointer to the new folder information.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER pFolder is NULL.

MSG_ERR_INVALID_FOLDER Mandatory fields are invalid.

MSG_ERR_STORAGE_FULL Storage is FULL.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.4 typedef MSG_ERROR_T(* MsgFwStoAddMessage)(MSG_MESSAGE_S *pMsg)

The prototype for adding a message into storage.

This function is to add a message common information into framework storage. Mandatory fields of message structure MUST be valid, otherwise the function will be failed. The mandatory fields for adding operation are msgId, parentId, msgType, storageId.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *pMsg* is a pointer to a MSG_MESSAGE_S struct.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER pMsg is NULL.

MSG_ERR_INVALID_MESSAGE The mandatory fields of message are invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_STORAGE_FULL Storage is FULL.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.5 typedef MSG_ERROR_T(* MsgFwStoAttachStorage)(MSG_STORAGE_ID_T StorageId)

The prototype for attaching a specific storage by storage id.

If the framework storage plugin uses the database, it can establish connection with the database and open the table in this function.

If the framework storage uses the files, it can open the files in this function.

For storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *StorageId* is an integer value indicating which storage.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_STORAGE_ERROR Storage error

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (*MSG_PLUGIN_ERROR_T*), which will be returned to application when *MsgGetStoragePluginError()* is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.6 `typedef MSG_ERROR_T(* MsgFwStoCheckStatus)(MSG_STORAGE_ID_T StorageId,
MSG_PLG_STORAGE_STATUS_T *pStatus)`

The prototype for checking the status of the storage by storage id.

Parameters:

← *StorageId* is an integer value indicating which storage.

→ *pStatus* indicates the status of the storage.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_NULL_POINTER *pStatus* is NULL.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (*MSG_PLUGIN_ERROR_T*), which will be returned to application when *MsgGetStoragePluginError()* is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.7 typedef MSG_ERROR_T(* MsgFwStoCountAccount)(const MSG_MESSAGE_TYPE_S *pMsgType, int *pCount)

The prototype for getting the accounts number of the specified message type.

This function is to get the number of accounts for the specified message type, such as SMS, MMS and Email. If the `pMsgType` is NULL, the function MUST return the count of all accounts. If the `pMsgType` is a specified type, the function MUST return account count of that type. If the `pMsgType` doesn't exist, `MSG_ERR_INVALID_MSG_TYPE` is returned.

Parameters:

- ← *pMsgType* is the message type to be counted.
- *pCount* is a pointer to the account number.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* `pCount` is NULL.
- MSG_ERR_INVALID_MSG_TYPE* The message type is not valid.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

`n` is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. `n` could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.8 typedef MSG_ERROR_T(* MsgFwStoCountMessage)(MSG_FOLDER_ID_T FolderId, bool bRecursive, const MSG_MESSAGE_TYPE_S *pMsgType, int *pCount)

The prototype for getting the number of messages under the specified folder.

Note:

If `pMsgType` is NULL, it MUST return the messages number of all the message types.

Parameters:

- ← *FolderId* indicates the folder to be counted.
- ← *bRecursive* indicates whether count the folder recursively.
- ← *pMsgType* points to the message type which should be counted.
- *pCount* is a integer pointer.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER `pCount` is NULL.

MSG_ERR_INVALID_MSG_TYPE Message type is invalid.

MSG_ERR_INVALID_FOLDER_ID Folder id is invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_PLUGIN+*n* the error code specific for plugin.

Note:

n is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.9 `typedef MSG_ERROR_T(* MsgFwStoCountSubFolder)(MSG_FOLDER_ID_T folderId, bool recursive, int *pCount)`

The prototype for getting the number of sub folders under the specified folder.

This function is to get the number of sub folders under the specified folder. It can do a recursive or non-recursive deleting operation based on the value of parameter `recursive`.

Parameters:

← ***folderId*** indicates the querying folder id.

← ***recursive*** indicates that this operation is recursive or non-recursive.

→ ***pCount*** is a pointer to the number of sub folders.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER `pCount` is NULL.

MSG_ERR_INVALID_FOLDER_ID This folder id doesn't exist in storage.

MSG_ERR_PLUGIN+*n* the error code specific for plugin.

Note:

n is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.10 typedef MSG_ERROR_T(* MsgFwStoCountUnreadMessage)(MSG_FOLDER_ID_T FolderId, bool bRecursive, const MSG_MESSAGE_TYPE_S *pMsgType, int *pCount)

The prototype for getting the number of unread messages under the specified folder.

Note:

If pMsgType is NULL, it MUST return the messages number of all the message types.

Parameters:

- ← *FolderId* indicates the folder to be counted.
- ← *bRecursive* indicates whether count the folder recursively.
- ← *pMsgType* points to the message type which should be counted.
- *pCount* is a integer pointer.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* pCount is NULL.
- MSG_ERR_INVALID_MSG_TYPE* Message type is invalid.
- MSG_ERR_INVALID_FOLDER_ID* Folder id is invalid.
- MSG_ERR_STORAGE_ERROR* Storage error.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.11 typedef MSG_ERROR_T(* MsgFwStoDelAllMsgInFolder)(MSG_FOLDER_ID_T FolderId)

Deletes all messages in the specified folder.

This function is to delete all messages in the specified folder, but it will not delete the subfolders and the messages in the subfolders.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

- ← *FolderId* is a folder Id.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_INVALID_FOLDER_ID Folder id is invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code ([MSG_PLUGIN_ERROR_T](#)), which will be returned to application when `MsgGetStoragePluginError()` is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.12 `typedef MSG_ERROR_T(* MsgFwStoDeleteAccount)(MSG_ACCOUNT_ID_T accountId)`

The prototype for deleting account common information.

This function is to delete the account common information in framework storage plugin. For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#).

Parameters:

← *accountId* identifies the specific account need to be delete.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_INVALID_ACCOUNT_ID This account is not exist.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code ([MSG_PLUGIN_ERROR_T](#)), which will be returned to application when `MsgGetStoragePluginError()` is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.13 `typedef MSG_ERROR_T(* MsgFwStoDeleteFolder)(MSG_FOLDER_ID_T folderId, bool recursive)`

The prototype for deleting a folder.

This function is to delete a folder in framework storage plugin. It can do a recursive or non-recursive deleting operation based on the value of parameter `recursive`. If the parameter `recursive` is false, the function MUST only delete this folder itself in case there is no content in this folder, otherwise `MSG_ERR_NOT_EMPTY_FOLDER` will be returned. If the parameter `recursive` is true, regardless of whether content is in this folder, the function will recursively delete all its sub folders, messages and this folder itself.

Parameters:

- ← *folderId* identifies the deleted folder.
- ← *recursive* indicates that the deleting operation is recursive or non-recursive.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_INVALID_FOLDER_ID* This folder id doesn't exist in storage.
- MSG_ERR_NOT_EMPTY_FOLDER* When `recursive` is false and the deleted folder has sub-folders or messages.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

`n` is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. `n` could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.14 `typedef MSG_ERROR_T(* MsgFwStoDeleteMessage)(MSG_MESSAGE_ID_T MsgId)`

The prototype for deleting a message.

This function is to delete a message common information in framework storage.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

- ← *MsgId* indicates to be deleted message's Id.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_INVALID_MESSAGE_ID Message id is invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.15 `typedef MSG_ERROR_T(* MsgFwStoDetachStorage)(MSG_STORAGE_ID_T StorageId)`

The prototype for detaching a specific storage by storage id.

If the framework storage plugin uses the database, it can close connection with the database in this function.

If the framework storage plugin uses the files, it can close the files in this function.

Parameters:

← *StorageId* is an integer value indicating which storage.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation

MSG_ERR_STORAGE_ERROR Storage error

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.16 `typedef MSG_ERROR_T(* MsgFwStoGetAccountIdsByRootId)(MSG_FOLDER_ID_T rootFolderId, MSG_ACCOUNT_ID_LIST_S *pAccountIdList)`

The prototype for getting account ids by root folder id.

This function is to get all the account ids whose root folder id is the specified folder id. Framework will guarantee the input folder id is root folder.

The memory of `accountIds` inside the parameter `accountIdList` MUST be allocated in this function. This function's caller MSUT release memory by invoking [MsgFwStoReleaseAccountIdList](#). However if this function is failed, the memory is NOT allocated in this function.

Parameters:

- ← *rootFolderId* is the querying root folder id.
- *pAccountIdList* is a pointer to the structure which contains account ids.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* `pAccountIdList` is NULL.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

`n` is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. `n` could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.17 `typedef MSG_ERROR_T(* MsgFwStoGetAccountIdsByStorage)(MSG_STORAGE_ID_T storageId, MSG_ACCOUNT_ID_LIST_S *pAccountIdList)`

The prototype for getting all the account ids on a specified storage.

This function is to get all account ids on a specified storage.

Parameters:

- ← *storageId* indicates the queried storage.
- *pAccountIdList* is a pointer to the account id list.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* `pAccountIdList` is NULL.
- MSG_ERR_INVALID_STORAGE* Storage id is invalid.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.18 `typedef MSG_ERROR_T(* MsgFwStoGetCommAccount)(MSG_ACCOUNT_ID_T accountId, MSG_ACCOUNT_S **ppAccount)`

The prototype for getting account common information.

This function is to get the account common information. The account common information includes all the fields of structure MSG_ACCOUNT_S except for `pExtData`. The memory of parameter `ppAccount` MUST be allocated in this function. This function's caller MSUT release memory by invoking [MsgFwStoReleaseCommAccount](#). However if this function is failed, the memory is NOT allocated in this function.

Parameters:

← *accountId* is the querying account id.

→ *ppAccount* is a pointer to a pointer to the structure which contains account common information.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER `ppAccount` is NULL.

MSG_ERR_INVALID_ACCOUNT_ID This account is not exist.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. *n* could be 0 if plugin doesn't want to specify the error.

For how to release framework account memory, please refer to [MsgFwStoReleaseCommAccount](#) .

Since:

1.0

6.2.2.19 `typedef MSG_ERROR_T(* MsgFwStoGetDefaultAccID)(const MSG_MESSAGE_TYPE_S *pMsgType, MSG_ACCOUNT_ID_T *pAccountId)`

The prototype for getting the default account id.

This function is to get the default account id for the specified message type. The message type can't be NULL.

Parameters:

- ← *pMsgType* is the specified message type.
- *pAccountId* returns the default account id.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* *pMsgType*, *pAccountId* are NULL.
- MSG_ERR_INVALID_MSG_TYPE* The message type is not valid.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (*MSG_PLUGIN_ERROR_T*), which will be returned to application when *MsgGetStoragePluginError()* is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.20 `typedef MSG_ERROR_T(* MsgFwStoGetFolder)(MSG_FOLDER_ID_T folderId, MSG_FOLDER_S **ppFolderInfo)`

The prototype for getting the folder information of the specified folder.

This function is to get all information of the specified folder, which includes common and extended information. For this folder structure may has extended structure, framework storage plugin MUST allocate folder structure memory in this function. This function's caller MUST release memory by invoking [MsgFwStoReleaseFolder](#). However if this function is failed, the memory is NOT allocated in this function.

Parameters:

- ← *folderId* indicates the querying folder id.
- *ppFolderInfo* is a pointer to a pointer to the folder detailed information structure.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* *ppFolderInfo* is NULL.
- MSG_ERR_INVALID_FOLDER_ID* This folder id doesn't exist in storage.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (*MSG_PLUGIN_ERROR_T*), which will be returned to application when *MsgGetStoragePluginError()* is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.21 `typedef MSG_ERROR_T(* MsgFwStoGetFolderByType)(MSG_FOLDER_ID_T rootFolderId, MSG_FOLDER_TYPE_T folderType, MSG_FOLDER_ID_LIST_S *pFolderIdList)`

The prototype for getting the folder by folder type for the specified root folder.

This function is to get the folder by folder type for the specified root folder. For example, it can retrieve inbox/outbox folders of a root folder.

Parameters:

- ← *rootFolderId* indicates the root folder.
- ← *folderType* indicates the searched folder type.
- *pFolderIdList* is a pointer to all the returned folder ids.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* *pFolderId* is NULL or *pFolderIdList* is NULL.
- MSG_ERR_INVALID_FOLDER_TYPE* Folder type is not valid for this operation.
- MSG_ERR_INVALID_FOLDER_ID* This folder id doesn't exist in storage.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (*MSG_PLUGIN_ERROR_T*), which will be returned to application when *MsgGetStoragePluginError()* is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.22 `typedef MSG_ERROR_T(* MsgFwStoGetFolderIdsByStorage)(MSG_STORAGE_ID_T storageId, MSG_FOLDER_ID_LIST_S *pFolderIdList)`

The prototype for getting all the folder id list on a specified storage.

This function is to get all folder ids on a specified storage.

Parameters:

- ← *storageId* indicates the queried storage.
- *pFolderIdList* is a pointer to the folder id list.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.

MSG_ERR_NULL_POINTER pFolderIdList is NULL.

MSG_ERR_INVALID_STORAGE Storage id is invalid.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.23 typedef MSG_ERROR_T(* MsgFwStoGetFolderParentId)(MSG_FOLDER_ID_T folderId, MSG_FOLDER_ID_T *pParentFolderId)

The prototype for getting the parent folder id of a folder.

This function is to get the parent folder id of a folder, and it MUST return -1 if it is a root folder.

Parameters:

← **folderId** indicates the querying folder id.

→ **pParentFolderId** is a pointer to the parent folder id.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER pParentFolderId is NULL.

MSG_ERR_INVALID_FOLDER_ID This folder id doesn't exist in storage.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.24 typedef MSG_ERROR_T(* MsgFwStoGetMessageIdsByStorage)(MSG_STORAGE_ID_T StorageId, MSG_MESSAGE_ID_LIST_S *pMsgIds)

Returns the message Ids by storage Id.

Applications need to allocate the structure pMsgIds.

When this function returns successfully, the storage ids will be filled in pMsgIds->msgIds. Applications need to call [MsgFwStoReleaseMessageList](#) to release the memory of pMsgIds->msgIds. However if this function is failed, the memory is NOT allocated in this function.

Parameters:

- ← *MsgId* indicates which message should be gotten.
- *pMsgIds* is a pointer to a MSG_MESSAGE_ID_LIST_S structure.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* pMsgIds is NULL.
- MSG_ERR_INVALID_STORAGE* Message id is invalid.
- MSG_ERR_STORAGE_ERROR* Storage error.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.25 `typedef MSG_ERROR_T(* MsgFwStoGetMessageParentId)(MSG_MESSAGE_ID_T MsgId, MSG_FOLDER_ID_T *pParentFolderId)`

The prototype for returning the folder id in which the message exists.

Parameters:

- ← *MsgId* indicates which message should be gotten.
- *pParentFolderId* a pointer to the parent folder id.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* pParentFolderId is NULL.
- MSG_ERR_INVALID_FOLDER_ID* Message id is invalid.
- MSG_ERR_STORAGE_ERROR* Storage error.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.26 `typedef MSG_ERROR_T(* MsgFwStoGetMsgCommInfo)(MSG_MESSAGE_ID_T MsgId, MSG_MESSAGE_S **ppMsg)`

The prototype for returning the message common information.

This function is to return the message common information including common extension data (`pCommExtData`) without specific extension data (`pExtData`).

Note:

Memory will be allocated inside this function.

Parameters:

- ← *MsgId* indicates which message should be gotten.
- *ppMsg* is a pointer to a message pointer.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* *ppMsg* is NULL.
- MSG_ERR_INVALID_MESSAGE_ID* Message id is invalid.
- MSG_ERR_STORAGE_ERROR* Storage error.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.27 `typedef MSG_ERROR_T(* MsgFwStoGetStorageStatus)(MSG_STORAGE_ID_T storageId, unsigned long long *totalSize, unsigned long long *usedSize, MSG_SIZE_UNIT_T *unit)`

Gets framework storage status by storage id.

This function is to get total size and used size of the specified framework storage.

Parameters:

- ← *storageId* is an integer value indicating which storage.
- *totalSize* the total size in this storage.
- *usedSize* used size in this storage.
- *unit* the unit of the size.

Returns:[MSG_ERROR_T](#)**Return values:***MSG_SUCCESS* Succeeds in operation.*MSG_ERR_NULL_POINTER* totalSize is NULL or usedSize is NULL or unit is NULL*MSG_ERR_INVALID_STORAGE* Storage id is invalid*MSG_ERR_PLUGIN+n* the error code specific for plugin.**Note:**

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.28 typedef MSG_ERROR_T(* MsgFwStoInitStorage)(MSG_STORAGE_ID_T StorageId)

The prototype for initializing the storage.

This function is to create all the data for framework storage plugin on the storage.

When this function returns successfully, all the necessary data will be created and be set with default values.

Parameters:

← *StorageId* is an integer value indicating which storage.

Returns:[MSG_ERROR_T](#)**Return values:***MSG_SUCCESS* Succeeds in operation, data on this storage will be reseted.*MSG_ERR_STORAGE_ERROR* Storage error.*MSG_ERR_INVALID_STORAGE* Storage id is invalid*MSG_ERR_STORAGE_FULL* Storage is FULL.*MSG_ERR_PLUGIN+n* the error code specific for plugin.**Note:**

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.29 `typedef MSG_ERROR_T(* MsgFwStoQueryAccount)(const MSG_MESSAGE_TYPE_S *pMsgType, MSG_ACCOUNT_ID_LIST_S *pAccountIdList)`

The prototype for querying the account id list of the specified message type.

This function is to get account id list for the specified message type. The parameter `pAccountIdList` is defined by the function caller, but the memory of its structure member `accountIds` MUST be allocated in this function. This function's caller MUST release the memory by invoking [MsgFwStoReleaseAccountIdList](#). However if this function is failed, the memory is NOT allocated in this function. If the `pMsgType` is NULL, the function will return all account id list. If the `pMsgType` is the specified type, the function will return account id list of that type. If the `pMsgType` doesn't exist, `MSG_ERR_INVALID_MSG_TYPE` is returned.

Parameters:

- ← *pMsgType* is the message type to be queried.
- *pAccountIdList* is a pointer to the account list.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* `pAccountIdList` is NULL.
- MSG_ERR_INVALID_MSG_TYPE* The message type is not valid.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

`n` is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. `n` could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.30 `typedef MSG_ERROR_T(* MsgFwStoQueryFolder)(MSG_FOLDER_ID_T folderId, int iBeginPos, int iCount, bool recursive, MSG_FOLDER_ID_LIST_S *pFolderIdList)`

The prototype for querying the sub folder ids under the specified folder.

This function is to query all the sub folder id list under the specified folder, and it can do a recursive or non-recursive searching based on the value of parameter `recursive`. The parameter `pFolderIdList` MUST be defined outside this function, but the memory of its member `folderIds` MUST be allocated in this function. So the function caller MUST release memory by invoking [MsgFwStoReleaseFolderIdList](#). However if this function is failed, the memory is NOT allocated in this function. If `folderId` is `MSG_INVALID_ID`, this function will query all the folders which have no parent folder.

Parameters:

- ← *folderId* indicates the folder id which is needed to query.
- ← *iBeginPos* is the start position of all the satisfied folders, it should be 0 or a positive number. If `iBeginPos` is negative, index begins from 0.

- ← *iCount* is the maximum number of folder information the caller wishes to receive. If *iCount* is negative, this method is to get all satisfied folders.
- ← *recursive* indicates whether this querying is recursive.
- *pFolderIdList* is the returned folder id list.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_NULL_POINTER* *pFolderIdList* is NULL.
- MSG_ERR_INVALID_FOLDER_ID* This folder id doesn't exist in storage and the folder id is not *MSG_INVALID_ID*.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code (*MSG_PLUGIN_ERROR_T*), which will be returned to application when *MsgGetStoragePluginError()* is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.31 `typedef MSG_ERROR_T(* MsgFwStoQueryMessage)(MSG_FOLDER_ID_T FolderId, bool bRecursive, int iBeginIndex, int iCount, const MSG_SORT_RULE_S *pSortRule, const MSG_QUERY_LIST_S *pQueryList, MSG_MESSAGE_ID_LIST_S *pMsgIdList)`

The prototype for getting the matched message ids under the specified folder.

If *pQueryList* is NULL, the function MUST return all messages in this folder. If *pSortRule* is NULL, the sort type is depended on underlying storage.

Parameters:

- ← *FolderId* indicates the folder to be queried.
- ← *bRecursive* indicates whether this searching is recursive.
- ← *iBeginIndex* indicates start position. If *iBeginIndex* is negative, index begins from 0.
- ← *iCount* is the maximum number of message information the caller wishes to receive. If *iCount* is negative, this method is to get all satisfied messages.
- ← *pSortRule* is the pointer to the sort rule.
- ← *pQueryList* is the pointer to the querylist. For query rule description, refer to [MSG_QUERY_LIST_S](#).
- *pMsgIdList* is a pointer to a *MSG_MESSAGE_ID_LIST_S* structure.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.
MSG_ERR_NULL_POINTER pMsgIdList is NULL.
MSG_ERR_INVALID_FOLDER_ID Folder id is invalid.
MSG_ERR_INVALID_SORT_TYPE Sort type is invalid.
MSG_ERR_INVALID_QUERY_ID Query id is invalid.
MSG_ERR_STORAGE_ERROR Storage error.
MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.32 `typedef void(* MsgFwStoReleaseAccountIdList)(MSG_ACCOUNT_ID_LIST_S *pAccountIdList)`

The prototype for release the memory of account id list.

This function is to release the memory of pointer `accountIds` inside `MSG_ACCOUNT_ID_LIST_S`. If `pAccountIdList` or `pAccountIdList->accountIds` is NULL, nothing happens. If `pAccountIdList` or `pAccountIdList->accountIds` is invalid, undefined behavior.

Note:

This function MUST NOT free this structure itself.

Parameters:

← *pAccountIdList* is a pointer to the account id list.

Since:

1.0

6.2.2.33 `typedef void(* MsgFwStoReleaseCommAccount)(MSG_ACCOUNT_S *pAccount)`

The prototype for releasing account common information.

This function is to release the memory of account common information. The memory of account common information is allocated in function [MsgFwStoGetCommAccount](#) .

If `pAccount` is NULL, nothing happens. If `pAccount` is invalid, undefined behavior.

Parameters:

← *pAccount* is a pointer to an account structure which contains account common information.

Since:

1.0

6.2.2.34 `typedef void(* MsgFwStoReleaseFolder)(MSG_FOLDER_S *pFolderInfo)`

The prototype for releasing the memory block of a folder.

This function is to release the memory of a folder, which is allocated in function [MsgFwStoGetFolder](#). For the folder may has extended data structure, this function is the exclusive way to release that folder memory.

If pFolderInfo is NULL, nothing happens. If pFolderInfo is invalid, undefined behavior.

Parameters:

← *pFolderInfo* is a pointer to the folder structure.

Example for creating folder [MsgFwStoGetFolder](#) .

Since:

1.0

6.2.2.35 `typedef void(* MsgFwStoReleaseFolderIdList)(MSG_FOLDER_ID_LIST_S *pFolderIdList)`

The prototype for releasing a memory block of folder id list.

This function is to release the memory of pointer `folderIds` inside `MSG_FOLDER_ID_LIST_S`. If pFolderIdList or pFolderIdList->folderIds is NULL, nothing happens. If pFolderIdList or pFolderIdList->folderIds is invalid, undefined behavior.

Note:

This function does NOT free this structure itself.

Parameters:

← *pFolderIdList* is a pointer to the folder id list.

Since:

1.0

6.2.2.36 `typedef void(* MsgFwStoReleaseMessageList)(MSG_MESSAGE_ID_LIST_S *pMsgIdList)`

The prototype for releasing the memory of the pointer in `MSG_MESSAGE_ID_LIST_S`.

This function is to release the memory of msgIds in [MSG_MESSAGE_ID_LIST_S](#) structure. If pMsgIdList or pMsgIdList->msgIds is NULL, nothing happens. If pMsgIdList or pMsgIdList->msgIds is invalid, undefined behavior.

This function is to release pMsgIdList->msgIds. It will not release pMsgIdList itself.

Parameters:

← *pMsgIdList* is a pointer to an `MSG_MESSAGE_ID_LIST_S` structure.

Since:

1.0

6.2.2.37 `typedef void(* MsgFwStoReleaseMsgCommon)(MSG_MESSAGE_S *pMsg)`

The prototype for releasing the memory of MSG_MESSAGE_S allocated in [MsgFwStoGetMsgCommInfo](#). If pMsg is NULL, nothing happens. If pMsg is invalid, undefined behavior.

Parameters:

← *pMsg* is a pointer to an MSG_MESSAGE_S structure.

Since:

1.0

6.2.2.38 `typedef MSG_ERROR_T(* MsgFwStoSetDefaultAccID)(const MSG_MESSAGE_TYPE_S *pMsgType, MSG_ACCOUNT_ID_T accountId)`

The prototype for setting the default account id.

This function is to set the default account id for the specified message type. The pMsgType cannot be NULL. Each message type has its own default account.

Parameters:

← *pMsgType* is the message type.

← *accountId* is used to set default account.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_INVALID_MSG_TYPE The message type is not valid.

MSG_ERR_INVALID_ACCOUNT_ID This account is not exist.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.39 `typedef MSG_ERROR_T(* MsgFwStoUpdateAccount)(const MSG_ACCOUNT_S *pAccount)`

The prototype for updating account common information.

The function is expected to update msgType, rootFolderId, storageId and defaultMsgStorageID for the account indentified by the given accountId as long as the given values are valid. Optionally, the function updates accountName, if necessary.

All account information except for `accountId` can be updated.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#).

If `pAccount->storageId` is different than original one, the account MUST be moved to updated storage.

Parameters:

← *pAccount* is a pointer to the data user defined to update the account.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER `pAccount` is NULL.

MSG_ERR_INVALID_ACCOUNT Some mandatory fields of account structure are not filled or not valid.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

`n` is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. `n` could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.40 `typedef MSG_ERROR_T(* MsgFwStoUpdateFolder)(const MSG_FOLDER_S *pFolder)`

The prototype for updating a folder.

This function is to update a folder in framework storage plugin. Following fields of folder structure MUST be valid, otherwise the function will be failed:

`folderId`, `parentId`, `storageId`. If the `sortType` is invalid, the default value `MSG_SORT_BY_UNKNOWN` MUST be set.

If `pFolder->parentId` is different than original one, the folder MUST be moved to updated destination folder.

If `pFolder->storageId` is different than original one, the folder MUST be moved to updated storage.

Parameters:

← *pFolder* is a pointer to a structure which contains all updated information.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER pFolder is NULL.

MSG_ERR_INVALID_FOLDER Mandatory fields are invalid.

MSG_ERR_INVALID_FOLDER_ID This folder id doesn't exist in storage.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.2.41 typedef MSG_ERROR_T(* MsgFwStoUpdateMessage)(const MSG_MESSAGE_S *pMsg)

The prototype for updating a message in storage.

The function is to update message common data for the message indentified by the given msgId as long as the given values are valid in framework storage.

Following fields of message structure MUST be valid, otherwise the function will be failed:

msgId, parentId, msgType, storageId.

If pMsg->parentId is different than original one, the message MUST be moved to updated destination folder.

If pMsg->storageId is different than original one, the message MUST be moved to updated storage.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *pMsg* is a const pointer to a MSG_MESSAGE_S structure.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER pMsg is NULL.

MSG_ERR_INVALID_MESSAGE The mandatory fields of message are invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.3 Function Documentation

6.2.3.1 `MSG_ERROR_T MsgFwStoCreateHandle (MSG_FW_STORAGE_S * hFwStoHandle)`

Creates a framework storage plugin handle.

This function is to provide framework storage plugin handle. This is the only way to get the framework storage plugin handle.

Parameters:

→ *hFwStoHandle* is the pointer to a framework storage plugin handle.

Returns:

`MSG_ERROR_T`

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER *hFwStoHandle* is NULL.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetSessionPluginError()` is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.2.3.2 `MSG_ERROR_T MsgFwStoDestroyHandle ()`

Destroys a framework storage plugin handle.

This function is to destroy a framework storage plugin handle. After destroying the handle, the handle can not be used anymore.

Returns:

`MSG_ERROR_T`

Return values:

MSG_SUCCESS Succeeds in operation.

Since:

1.0

6.3 Msg_Mapi_Account.h File Reference

6.3.1 Detailed Description

Defines the account structure.

```
#include <stdlib.h>
#include "Msg_Mapi_Types.h"
#include "Msg_Mapi_Folder.h"
```

Data Structures

- struct [_MSG_ACCOUNT_S](#)

Represents an account.

Typedefs

- typedef struct [_MSG_ACCOUNT_S](#) MSG_ACCOUNT_S

6.3.2 Typedef Documentation

6.3.2.1 typedef struct _MSG_ACCOUNT_S MSG_ACCOUNT_S

6.4 Msg_Mapi_Contact.h File Reference

6.4.1 Detailed Description

Defines the contact structure.

```
#include <stdlib.h>
#include "Msg_Mapi_Types.h"
```

Data Structures

- struct [_MSG_CONTACT_S](#)
Represents a contact.
- struct [_MSG_CONTACT_LIST_S](#)
Represents a contact list.

Typedefs

- typedef struct [_MSG_CONTACT_S](#) MSG_CONTACT_S
- typedef struct [_MSG_CONTACT_LIST_S](#) MSG_CONTACT_LIST_S

6.4.2 Typedef Documentation

6.4.2.1 typedef struct _MSG_CONTACT_LIST_S MSG_CONTACT_LIST_S

6.4.2.2 typedef struct _MSG_CONTACT_S MSG_CONTACT_S

6.5 Msg_Mapi_FilterRule.h File Reference

6.5.1 Detailed Description

Defines the filter related structures.

MSG_FILTER_S and MSG_FILTER_LIST_S define the criteria for session's interested incoming messages.

MSG_FILTER_S represents the basic filter unit, with a pair of filter type and filter value. A filter is satisfied when the filter value meets the requirement of specific filter type.

MSG_FILTER_LIST_S is composed of a series of various MSG_FILTER_S. A filter list is satisfied when one of filters in the list is satisfied.

The first filter in the list has the highest priority, the last the lowest. Framework will check the filter list from the first to the last. Once an incoming message satisfies one filter in the list, framework would stop checking and return.

The framework sends the satisfying messages to filter-list associated applications. If the intercept is set true, the lower priority session would not receive the messages.

See [MsgPlgFilterCheckRules](#).

```
#include "Msg_Mapi_Types.h"
```

Data Structures

- struct [_MSG_FILTER_ID_LIST_S](#)
Represents a filter id list.
- struct [_MSG_FILTER_S](#)
Represents a filter.
- struct [_MSG_FILTER_LIST_S](#)
Represents a filter list.

Typedefs

- typedef int [MSG_FILTER_ID_T](#)
Represents the filter Id.
- typedef struct [_MSG_FILTER_ID_LIST_S](#) [MSG_FILTER_ID_LIST_S](#)
- typedef struct [_MSG_FILTER_S](#) [MSG_FILTER_S](#)
- typedef struct [_MSG_FILTER_LIST_S](#) [MSG_FILTER_LIST_S](#)

Enumerations

- enum [_MSG_FILTER_ID_E](#) {
[MSG_FILTER_BY_UNKNOWN](#) = 0, [MSG_FILTER_BY_MSG_TYPE](#), [MSG_FILTER_BY_-FROM](#), [MSG_FILTER_BY_HAS_ATTACHMENT](#),

```
MSG_FILTER_BY_HAS_DRM_CONTENT, MSG_FILTER_BY_SIZE_GT, MSG_FILTER_BY_SIZE_LT, MSG_FILTER_BY_PRIORITY_GT,
```

```
MSG_FILTER_BY_PRIORITY_LT, MSG_FILTER_BY_USER_DEF_BEGIN = 0x00001000 }
```

Represents the values of filter id.

6.5.2 Typedef Documentation

6.5.2.1 typedef struct _MSG_FILTER_ID_LIST_S MSG_FILTER_ID_LIST_S

6.5.2.2 typedef int MSG_FILTER_ID_T

Represents the filter Id.

A filter Id represents a unique filter type. The values for this type SHOULD be in [_MSG_FILTER_ID_E](#) or other extended filter Ids.

6.5.2.3 typedef struct _MSG_FILTER_LIST_S MSG_FILTER_LIST_S

6.5.2.4 typedef struct _MSG_FILTER_S MSG_FILTER_S

6.5.3 Enumeration Type Documentation

6.5.3.1 enum _MSG_FILTER_ID_E

Represents the values of filter id.

This enum is used as the value of [MSG_FILTER_ID_T](#).

For example:

```
MSG_FILTER_S filter;
filter.filterId = MSG_FILTER_BY_MSG_TYPE;
filter.filterValue = (unsigned char*)malloc(sizeof(MSG_MESSAGE_TYPE_S));

MSG_MESSAGE_TYPE_S* msgType = (MSG_MESSAGE_TYPE_S*)filter.filterValue;
msgType->mainType = MSG_MESSAGE_TYPE_SMS;
msgType->subType = MSG_MESSAGE_TYPE_SMS_TEXT;

//set the filter
...
...
//the session will only receive SMS.
```

Enumerator:

MSG_FILTER_BY_UNKNOWN Unknown sort type.

MSG_FILTER_BY_MSG_TYPE The messages whose message type are filterValue will be filtered out. filterValue should be MSG_MESSAGE_TYPE_S type.

MSG_FILTER_BY_FROM The messages that come from filterValue will be filtered out. filterValue should be char * type, which is encoded in ASCII.

MSG_FILTER_BY_HAS_ATTACHMENT Filter by whether the message has attachment.

If filterValue is set 0, the messages that do not has attachment will be filtered out.

If filterValue is set 1, the messages that have attachment will be filtered out.

MSG_FILTER_BY_HAS_DRM_CONTENT Filter by whether the message has DRM content.

If `filterValue` is set 0, the messages that do not has DRM content will be filtered out.

If `filterValue` is set 1, the messages that have DRM content will be filtered out.

MSG_FILTER_BY_SIZE_GT The messages whose size are greater than or equal the `filterValue` will be filtered out. `filterValue` should be `size_t` type

MSG_FILTER_BY_SIZE_LT The messages whose size are less than or equal the `filterValue` will be filtered out. `filterValue` should be `size_t` type

MSG_FILTER_BY_PRIORITY_GT The messages whose priority are greater than or equal the `filterValue` will be filtered out. `filterValue` should be `MSG_MESSAGE_PRIORITY_T` type.

MSG_FILTER_BY_PRIORITY_LT The messages whose priority are less than or equal the `filterValue` will be filtered out. `filterValue` should be `MSG_MESSAGE_PRIORITY_T` type.

MSG_FILTER_BY_USER_DEF_BEGIN User defined filter type begin value.

6.6 Msg_Mapi_Folder.h File Reference

6.6.1 Detailed Description

Defines the folder related structures.

For the detail information of folder , please refer to “Folder” in *Messaging API* [2] .

```
#include <stdlib.h>
#include "Msg_Mapi_Types.h"
#include "Msg_Mapi_QueryRule.h"
```

Data Structures

- struct [_MSG_FOLDER_S](#)

Represents a folder.

Typedefs

- typedef struct [_MSG_FOLDER_S](#) MSG_FOLDER_S

6.6.2 Typedef Documentation

6.6.2.1 typedef struct _MSG_FOLDER_S MSG_FOLDER_S

6.7 Msg_Mapi_Message.h File Reference

6.7.1 Detailed Description

Defines the message structure.

```
#include <stdlib.h>
#include "Msg_Mapi_Types.h"
#include "Msg_Mapi_Contact.h"
#include "time.h"
```

Data Structures

- struct [_MSG_MESSAGE_S](#)

Represents a message.

Typedefs

- typedef struct [_MSG_MESSAGE_S](#) [MSG_MESSAGE_S](#)

6.7.2 Typedef Documentation

6.7.2.1 typedef struct _MSG_MESSAGE_S MSG_MESSAGE_S

6.8 Msg_Mapi_Message_Request.h File Reference

6.8.1 Detailed Description

Defines the message request structure.

```
#include <stdlib.h>
#include "Msg_Mapi_Types.h"
#include "Msg_Mapi_Message.h"
```

Data Structures

- struct [_MSG_MESSAGE_REQUEST_S](#)
Represents a request that will be sent to messaging framework.

Typedefs

- typedef struct [_MSG_MESSAGE_REQUEST_S](#) MSG_MESSAGE_REQUEST_S

6.8.2 Typedef Documentation

6.8.2.1 typedef struct _MSG_MESSAGE_REQUEST_S MSG_MESSAGE_REQUEST_S

6.9 Msg_Mapi_Message_Status.h File Reference

6.9.1 Detailed Description

Defines the message status structure.

```
#include "stdlib.h"
#include "Msg_Mapi_Types.h"
```

Data Structures

- struct [_MSG_MESSAGE_STATUS_S](#)
Represents a message status.

Typedefs

- typedef struct [_MSG_MESSAGE_STATUS_S](#) MSG_MESSAGE_STATUS_S

6.9.2 Typedef Documentation

6.9.2.1 typedef struct _MSG_MESSAGE_STATUS_S MSG_MESSAGE_STATUS_S

6.10 Msg_Mapi_QueryRule.h File Reference

6.10.1 Detailed Description

Defines the query related structure.

Query rule is used to query some specific messages from storage. Query rule list is composed of various query rules. Once the message satisfies one of the query rules in the list, it satisfies the query rule list.

Applications should assemble a query rule list to filter the message even there is only one query rule. Please refer to [MSG_QUERY_LIST_S](#)

Framework storage plugin SHOULD implement the query function and provide the supported queryId and queryValue for user.

```
#include "Msg_Mapi_Types.h"
```

Data Structures

- struct [_MSG_QUERY_ID_LIST_S](#)
Represents a query Id list.
- struct [_MSG_QUERY_S](#)
Represents a query rule.
- struct [_MSG_QUERY_LIST_S](#)
Represents a query list.
- struct [_MSG_SORT_TYPE_LIST_S](#)
Represents a sort type list.
- struct [_MSG_SORT_RULE_S](#)
Represents a sort rule.

Typedefs

- typedef int [MSG_QUERY_ID_T](#)
The query id.
- typedef struct [_MSG_QUERY_ID_LIST_S](#) [MSG_QUERY_ID_LIST_S](#)
- typedef struct [_MSG_QUERY_S](#) [MSG_QUERY_S](#)
- typedef struct [_MSG_QUERY_LIST_S](#) [MSG_QUERY_LIST_S](#)
- typedef int [MSG_SORT_TYPE_T](#)
Represents a sort type.
- typedef struct [_MSG_SORT_TYPE_LIST_S](#) [MSG_SORT_TYPE_LIST_S](#)
- typedef struct [_MSG_SORT_RULE_S](#) [MSG_SORT_RULE_S](#)

Enumerations

- enum `_MSG_QUERY_ID_E` {
`MSG_QUERY_BY_UNKNOWN` = 0, `MSG_QUERY_BY_BEFORE_DATETIME`, `MSG_QUERY_BY_AFTER_DATETIME`, `MSG_QUERY_BY_FROM`,
`MSG_QUERY_BY_TO`, `MSG_QUERY_BY_SUBJECT`, `MSG_QUERY_BY_SIZE`, `MSG_QUERY_BY_STATUS`,
`MSG_QUERY_BY_DELIVERY_STATUS`, `MSG_QUERY_BY_HASATTACHMENT`, `MSG_QUERY_BY_HASDRMCONTENT`, `MSG_QUERY_BY_PRIORITY`,
`MSG_QUERY_BY_USER_DEF_BEGIN` = 0x00001000 }

Represents the values of query Id.

- enum `_MSG_SORT_TYPE_E` {
`MSG_SORT_BY_UNKNOWN` = 0, `MSG_SORT_BY_DISPLAY_SUBJECT`, `MSG_SORT_BY_DISPLAY_FROM`, `MSG_SORT_BY_DISPLAY_TO`,
`MSG_SORT_BY_DISPLAY_TIME`, `MSG_SORT_BY_SIZE`, `MSG_SORT_BY_STATUS`, `MSG_SORT_BY_DELIVERY_STATUS`,
`MSG_SORT_BY_HASATTACHMENT`, `MSG_SORT_BY_HASDRMCONTENT`, `MSG_SORT_BY_PRIORITY`, `MSG_SORT_BY_USER_DEF_BEGIN` = 0x00001000 }

Represents the values of sort types.

6.10.2 Typedef Documentation

6.10.2.1 `typedef struct _MSG_QUERY_ID_LIST_S MSG_QUERY_ID_LIST_S`

6.10.2.2 `typedef int MSG_QUERY_ID_T`

The query id.

One query id presents a unique query type.

6.10.2.3 `typedef struct _MSG_QUERY_LIST_S MSG_QUERY_LIST_S`

6.10.2.4 `typedef struct _MSG_QUERY_S MSG_QUERY_S`

6.10.2.5 `typedef struct _MSG_SORT_RULE_S MSG_SORT_RULE_S`

6.10.2.6 `typedef struct _MSG_SORT_TYPE_LIST_S MSG_SORT_TYPE_LIST_S`

6.10.2.7 `typedef int MSG_SORT_TYPE_T`

Represents a sort type.

The values for this type SHOULD be in `_MSG_SORT_TYPE_E`

6.10.3 Enumeration Type Documentation

6.10.3.1 enum _MSG_QUERY_ID_E

Represents the values of query Id.

This enum is used as the value of [MSG_QUERY_ID_T](#), which is used as an input parameter in method `MsgQueryMessage` (see *Messaging API* [2]). Framework storage plugin can extend the supported query Id in this enum. Applications can use the query Id when querying messages.

Enumerator:

MSG_QUERY_BY_UNKNOWN Unknown query Id
MSG_QUERY_BY_BEFORE_DATETIME Query message whose time is before or equal to the datetime.
MSG_QUERY_BY_AFTER_DATETIME Query message whose time is after or equal to the date-time
MSG_QUERY_BY_FROM Query message by from_keyword
MSG_QUERY_BY_TO Query message by to_keyword
MSG_QUERY_BY_SUBJECT Query message by subject_keyword
MSG_QUERY_BY_SIZE Query message by size
MSG_QUERY_BY_STATUS Query message by status
MSG_QUERY_BY_DELIVERY_STATUS Query message by delivery status
MSG_QUERY_BY_HASATTACHMENT Query message by has attachment
MSG_QUERY_BY_HASDRMCONTENT Query message by drm content
MSG_QUERY_BY_PRIORITY Query message by priority
MSG_QUERY_BY_USER_DEF_BEGIN User defined query Id begin value

6.10.3.2 enum _MSG_SORT_TYPE_E

Represents the values of sort types.

This enum is used as the value of [MSG_SORT_TYPE_T](#).

Enumerator:

MSG_SORT_BY_UNKNOWN Unknown sort type
MSG_SORT_BY_DISPLAY_SUBJECT Sort by display subject
MSG_SORT_BY_DISPLAY_FROM Sort by display from
MSG_SORT_BY_DISPLAY_TO Sort by display to
MSG_SORT_BY_DISPLAY_TIME Sort by display time
MSG_SORT_BY_SIZE Sort by size
MSG_SORT_BY_STATUS Sort by status
MSG_SORT_BY_DELIVERY_STATUS Sort by delivery status
MSG_SORT_BY_HASATTACHMENT Sort by has attachment
MSG_SORT_BY_HASDRMCONTENT Sort by drm content
MSG_SORT_BY_PRIORITY Sort by priority
MSG_SORT_BY_USER_DEF_BEGIN User defined sort type begin value

6.11 Msg_Mapi_Types.h File Reference

6.11.1 Detailed Description

Defines general types.

Data Structures

- struct [_MSG_MESSAGE_TYPE_S](#)
Represents a message type.
- struct [_MSG_MESSAGE_TYPE_LIST_S](#)
Represents a message types list.
- struct [_MSG_STORAGE_ID_LIST_S](#)
Represents a storage id list.
- struct [_MSG_STRING_S](#)
Represents a string.
- struct [_MSG_MESSAGE_ID_LIST_S](#)
Represents a message id list.
- struct [_MSG_FOLDER_ID_LIST_S](#)
Represents a folder id list.
- struct [_MSG_ACCOUNT_ID_LIST_S](#)
Represents an account id list.

Defines

- #define [bool](#) unsigned char
Defines bool type.
- #define [true](#) 1
Defines true value.
- #define [false](#) 0
Defines false value.
- #define [MSG_INVALID_MESSAGE_ID](#) 0xffffffff
Defines invalid id for message.
- #define [MSG_INVALID_ACCOUNT_ID](#) 0xffff
Defines invalid id for account.
- #define [MSG_INVALID_FOLDER_ID](#) 0xffff
Defines invalid id for folder.

- #define [MSG_DEFAULT_ACCOUNT_ID](#) 0
Defines default account id.
- #define [MSG_DEFAULT_FOLDER_ID](#) 0
Defines default folder id.
- #define [MSG_ERROR_BEGIN](#) (-255)
Defines begin value for error.

Typedefs

- typedef int [MSG_MAIN_TYPE_T](#)
Represents a message main type.
- typedef int [MSG_SUB_TYPE_T](#)
Represents a message sub type.
- typedef struct [_MSG_MESSAGE_TYPE_S](#) [MSG_MESSAGE_TYPE_S](#)
- typedef struct [_MSG_MESSAGE_TYPE_LIST_S](#) [MSG_MESSAGE_TYPE_LIST_S](#)
- typedef int [MSG_ERROR_T](#)
Represents an error code.
- typedef unsigned int [MSG_PLUGIN_ERROR_T](#)
Error code specific for plugins.
- typedef int [MSG_SESSION_PRIORITY_T](#)
Represents a session priority.
- typedef int [MSG_MESSAGE_PRIORITY_T](#)
Represents a message priority.
- typedef int [MSG_STATUS_T](#)
Represents a status.
- typedef int [MSG_REQUEST_TYPE_T](#)
Represents a request type.
- typedef unsigned int [MSG_REQUEST_ID_T](#)
Represents a request id, which is unique for each request.
- typedef unsigned int [MSG_STORAGE_ID_T](#)
Represents a Storage id.
- typedef unsigned int [MSG_MEDIA_TYPE_T](#)
Represents a media type.
- typedef unsigned int [MSG_FOLDER_TYPE_T](#)

Represents a folder type.

- typedef struct [_MSG_STORAGE_ID_LIST_S](#) MSG_STORAGE_ID_LIST_S
- typedef int [MSG_CONTACT_TYPE_T](#)

Represents a contact type.

- typedef int [MSG_ENCODE_TYPE_T](#)

Represents an encoding type.

- typedef struct [_MSG_STRING_S](#) MSG_STRING_S
- typedef unsigned long [MSG_MESSAGE_ID_T](#)

Represents a message id.

- typedef unsigned int [MSG_FOLDER_ID_T](#)

Represents a folder id.

- typedef unsigned int [MSG_ACCOUNT_ID_T](#)

Represents an account id.

- typedef struct [_MSG_MESSAGE_ID_LIST_S](#) MSG_MESSAGE_ID_LIST_S
- typedef struct [_MSG_FOLDER_ID_LIST_S](#) MSG_FOLDER_ID_LIST_S
- typedef struct [_MSG_ACCOUNT_ID_LIST_S](#) MSG_ACCOUNT_ID_LIST_S
- typedef int [MSG_SIZE_UNIT_T](#)

Represents the unit of the size.

Enumerations

- enum [_MSG_MAIN_TYPE_E](#) {
[MSG_MESSAGE_TYPE_UNKNOWN](#), [MSG_MESSAGE_TYPE_MMS](#), [MSG_MESSAGE_TYPE_EMAIL](#), [MSG_MESSAGE_TYPE_SMS](#),
[MSG_MESSAGE_TYPE_IMODE](#) }

Represents the values of message main types.

- enum [_MSG_ERROR_E](#) {
[MSG_SUCCESS](#) = 0, [MSG_ERR_FS_ERROR](#) = MSG_ERROR_BEGIN - 1, [MSG_ERR_FS_FULL](#) = MSG_ERROR_BEGIN - 2, [MSG_ERR_INVALID_STORAGE](#) = MSG_ERROR_BEGIN - 3,
[MSG_ERR_INVALID_MSG_TYPE](#) = MSG_ERROR_BEGIN - 4, [MSG_ERR_INVALID_SORT_TYPE](#) = MSG_ERROR_BEGIN - 5, [MSG_ERR_INVALID_FOLDER_ID](#) = MSG_ERROR_BEGIN - 6, [MSG_ERR_INVALID_STORAGE_REG](#) = MSG_ERROR_BEGIN - 7,
[MSG_ERR_INVALID_CONNECTION](#) = MSG_ERROR_BEGIN - 8, [MSG_ERR_NULL_POINTER](#) = MSG_ERROR_BEGIN - 9, [MSG_ERR_INVALID_MESSAGE_ID](#) = MSG_ERROR_BEGIN - 10, [MSG_ERR_INVALID_ACCOUNT_ID](#) = MSG_ERROR_BEGIN - 11,
[MSG_ERR_INVALID_QUERY_ID](#) = MSG_ERROR_BEGIN - 12, [MSG_ERR_SESSION_NOT_CONNECTED](#) = MSG_ERROR_BEGIN - 13, [MSG_ERR_INVALID_SESSION_HANDLE](#) = MSG_ERROR_BEGIN - 14, [MSG_ERR_PERMISSION_DENIED](#) = MSG_ERROR_BEGIN - 15,

```

MSG_ERR_STORAGE_NOT_INIT = MSG_ERROR_BEGIN - 16, MSG_ERR_STORAGE_
REG_NOT_INIT = MSG_ERROR_BEGIN - 17, MSG_ERR_INVALID_SESSION_PRIORITY =
MSG_ERROR_BEGIN - 18, MSG_ERR_INVALID_FILTER_ID = MSG_ERROR_BEGIN - 19,
MSG_ERR_INVALID_REQUEST = MSG_ERROR_BEGIN - 20, MSG_ERR_INVALID_
FOLDER_TYPE = MSG_ERROR_BEGIN - 21, MSG_ERR_INVALID_FOLDER = MSG_
ERROR_BEGIN - 22, MSG_ERR_NOT_EMPTY_FOLDER = MSG_ERROR_BEGIN - 23,
MSG_ERR_INVALID_ACCOUNT = MSG_ERROR_BEGIN - 24, MSG_ERR_CONNECTION_
NOT_CONNECT = MSG_ERROR_BEGIN - 25, MSG_ERR_INVALID_MESSAGE = MSG_
ERROR_BEGIN - 26, MSG_ERR_STORAGE_FULL = MSG_ERROR_BEGIN - 27,
MSG_ERR_ROOTFOLDER_INUSE = MSG_ERROR_BEGIN - 28, MSG_ERR_STORAGE_
ERROR = MSG_ERROR_BEGIN - 29, MSG_ERR_MEMORY_ERROR = MSG_ERROR_BEGIN
- 30, MSG_ERR_COMMUNICATION_ERROR = MSG_ERROR_BEGIN - 31,
MSG_ERR_MAX_NUMBER_REACHED = MSG_ERROR_BEGIN - 32, MSG_ERR_
INVALID_REQUEST_ID = MSG_ERROR_BEGIN - 33, MSG_ERR_UNKNOWN = MSG_
ERROR_BEGIN - 0x00008000, MSG_ERR_FRAMEWORK_RESERVE_BEGIN = MSG_
ERROR_BEGIN - 0x00010000,
MSG_ERR_FRAMEWORK_RESERVE_END = MSG_ERROR_BEGIN - 0x00020000, MSG_
ERR_PLUGIN = MSG_ERROR_BEGIN - 0x08000000, MSG_ERR_PLUGIN_DEF_BEGIN_
VALUE = MSG_ERR_PLUGIN }

```

Represents the values of error codes.

- enum `_MSG_SESSION_PRIORITY_E` {
`MSG_SESSION_PRIORITY_LOWEST`, `MSG_SESSION_PRIORITY_LOW`, `MSG_SESSION_`
`PRIORITY_NORMAL`, `MSG_SESSION_PRIORITY_HIGH`,
`MSG_SESSION_PRIORITY_HIGHEST` }

Represents the values of session priority.

- enum `_MSG_MESSAGE_PRIORITY_E` { `MSG_MESSAGE_PRIORITY_LOW`, `MSG_`
`MESSAGE_PRIORITY_NORMAL`, `MSG_MESSAGE_PRIORITY_HIGH` }

Represents the values of message priority.

- enum `_MSG_STATUS_E` {
`MSG_STATUS_UNKNOWN` = 0, `MSG_STATUS_READ`, `MSG_STATUS_UNREAD`, `MSG_`
`STATUS_REPLIED`,
`MSG_STATUS_FORWARDED`, `MSG_STATUS_DRAFT`, `MSG_STATUS_EDITING`, `MSG_`
`STATUS_PROTECTED`,
`MSG_STATUS_SENDING`, `MSG_STATUS_SENT`, `MSG_STATUS_FAIL_TO_SEND`, `MSG_`
`STATUS_SAVED_NOTSENT`,
`MSG_STATUS_SENT_NOTSAVED`, `MSG_STATUS_SENT_SAVED`, `MSG_STATUS_FAIL_`
`TO_SENDSAVE`, `MSG_STATUS_REQ_CANCELLED`,
`MSG_STATUS_USER_DEF_BEGIN` = 0x08000000 }

Represents the values of status.

- enum `_MSG_REQUEST_TYPE_E` {
`MSG_REQUEST_TYPE_INVALID` = 0, `MSG_REQUEST_TYPE_RESERVED_BEGIN`, `MSG_`
`REQUEST_TYPE_SEND`, `MSG_REQUEST_TYPE_SENDSAVE`,
`MSG_REQUEST_TYPE_RESERVED_END` = `MSG_REQUEST_TYPE_RESERVED_BEGIN` +
0x00001000, `MSG_REQUEST_TYPE_USER_DEF_BEGIN` = 0x08000000 }

Represents the values of request types.

- enum `_MSG_STORAGE_ID_VALUE_E` { `MSG_STORAGE_UNKNOWN` = 0, `MSG_STORAGE_DEFAULT` }
- enum `_MSG_MEDIA_TYPE_E` { `MSG_MEDIA_FLASH`, `MSG_MEDIA_REMOVABLE`, `MSG_MEDIA_SIM` }

Represents the values of media types.

- enum `_MSG_FOLDER_TYPE_E` {
`MSG_FOLDER_TYPE_ROOT`, `MSG_FOLDER_TYPE_INBOX`, `MSG_FOLDER_TYPE_OUTBOX`, `MSG_FOLDER_TYPE_DRAFTS`,
`MSG_FOLDER_TYPE_SENT`, `MSG_FOLDER_TYPE_DELETED`, `MSG_FOLDER_TYPE_USERDEFINE` }

Represents the values of folder types.

- enum `_MSG_CONTACT_TYPE_E` { `MSG_CONTACT_TYPE_UNKNOWN` = 0, `MSG_CONTACT_TYPE_EMAIL`, `MSG_CONTACT_TYPE_PLMN` }

Represents the values of Contact types.

- enum `_MSG_ENCODE_TYPE_E` {
`MSG_ENCODE_UTF8`, `MSG_ENCODE_UTF16`, `MSG_ENCODE_ASCII`, `MSG_ENCODE_GSM7BIT`,
`MSG_ENCODE_UCS2`, `MSG_ENCODE_SHIFT_JIS` }

Represents the values of String encoding types.

- enum `_MSG_SIZE_UNIT_E` {
`MSG_SIZE_UNIT_BYTE`, `MSG_SIZE_UNIT_KBYTE`, `MSG_SIZE_UNIT_MBYTE`, `MSG_SIZE_UNIT_GBYTE`,
`MSG_SIZE_UNIT_NUMBER` }

6.11.2 Define Documentation

6.11.2.1 #define bool unsigned char

Defines bool type.

6.11.2.2 #define false 0

Defines false value.

6.11.2.3 #define MSG_DEFAULT_ACCOUNT_ID 0

Defines default account id.

6.11.2.4 #define MSG_DEFAULT_FOLDER_ID 0

Defines default folder id.

6.11.2.5 #define MSG_ERROR_BEGIN (-255)

Defines begin value for error.

6.11.2.6 #define MSG_INVALID_ACCOUNT_ID 0xffff

Defines invalid id for account.

6.11.2.7 #define MSG_INVALID_FOLDER_ID 0xffff

Defines invalid id for folder.

6.11.2.8 #define MSG_INVALID_MESSAGE_ID 0xffffffff

Defines invalid id for message.

6.11.2.9 #define true 1

Defines true value.

6.11.3 Typedef Documentation**6.11.3.1 typedef struct _MSG_ACCOUNT_ID_LIST_S MSG_ACCOUNT_ID_LIST_S****6.11.3.2 typedef unsigned int MSG_ACCOUNT_ID_T**

Represents an account id.

6.11.3.3 typedef int MSG_CONTACT_TYPE_T

Represents a contact type.

The values for this type SHOULD be in [_MSG_CONTACT_TYPE_E](#)

6.11.3.4 typedef int MSG_ENCODE_TYPE_T

Represents an encoding type.

The values for this type SHOULD be in [_MSG_ENCODE_TYPE_E](#)

6.11.3.5 typedef int MSG_ERROR_T

Represents an error code.

The values for this type SHOULD be in [_MSG_ERROR_E](#)

6.11.3.6 typedef struct _MSG_FOLDER_ID_LIST_S MSG_FOLDER_ID_LIST_S**6.11.3.7 typedef unsigned int MSG_FOLDER_ID_T**

Represents a folder id.

6.11.3.8 typedef unsigned int MSG_FOLDER_TYPE_T

Represents a folder type.

The values for this type SHOULD be in [_MSG_FOLDER_TYPE_E](#)

6.11.3.9 typedef int MSG_MAIN_TYPE_T

Represents a message main type.

The values for this type SHOULD be in [_MSG_MAIN_TYPE_E](#)

6.11.3.10 typedef unsigned int MSG_MEDIA_TYPE_T

Represents a media type.

The values for this type SHOULD be in [_MSG_MEDIA_TYPE_E](#)

6.11.3.11 typedef struct _MSG_MESSAGE_ID_LIST_S MSG_MESSAGE_ID_LIST_S**6.11.3.12 typedef unsigned long MSG_MESSAGE_ID_T**

Represents a message id.

6.11.3.13 typedef int MSG_MESSAGE_PRIORITY_T

Represents a message priority.

The values for this type SHOULD be in [_MSG_MESSAGE_PRIORITY_E](#)

6.11.3.14 typedef struct _MSG_MESSAGE_TYPE_LIST_S MSG_MESSAGE_TYPE_LIST_S**6.11.3.15 typedef struct _MSG_MESSAGE_TYPE_S MSG_MESSAGE_TYPE_S****6.11.3.16 typedef unsigned int MSG_PLUGIN_ERROR_T**

Error code specific for plugins.

6.11.3.17 typedef unsigned int MSG_REQUEST_ID_T

Represents a request id, which is unique for each request.

For details of request, please refer to “Request” in *Messaging API* [2] .

6.11.3.18 typedef int MSG_REQUEST_TYPE_T

Represents a request type.

The values for this type SHOULD be in [_MSG_REQUEST_TYPE_E](#)

6.11.3.19 typedef int MSG_SESSION_PRIORITY_T

Represents a session priority.

The values for this type SHOULD be in [_MSG_SESSION_PRIORITY_E](#)

6.11.3.20 typedef int MSG_SIZE_UNIT_T

Represents the unit of the size.

The values for this type SHOULD be in [_MSG_SIZE_UNIT_E](#)

6.11.3.21 typedef int MSG_STATUS_T

Represents a status.

The values for this type SHOULD be in [_MSG_STATUS_E](#)

6.11.3.22 typedef struct _MSG_STORAGE_ID_LIST_S MSG_STORAGE_ID_LIST_S

6.11.3.23 typedef unsigned int MSG_STORAGE_ID_T

Represents a Storage id.

For details please refer to “Storage” in *Messaging API* [2] .

6.11.3.24 typedef struct _MSG_STRING_S MSG_STRING_S

6.11.3.25 typedef int MSG_SUB_TYPE_T

Represents a message sub type.

Each main type message can be divided into some sub types. For instance of SMS, the sub types can be CLASS0 SMS, CLASS2 SMS and so on.

6.11.4 Enumeration Type Documentation

6.11.4.1 enum _MSG_CONTACT_TYPE_E

Represents the values of Contact types.

When MSG_CONTACT_TYPE_EMAIL is used, the contact value is an email address; When MSG_CONTACT_TYPE_PLMN is used, the contact value is a mobile number.

Enumerator:

MSG_CONTACT_TYPE_UNKNOWN The contact type is unknown

MSG_CONTACT_TYPE_EMAIL The contact type is for email address like `abc@example.comemail`

MSG_CONTACT_TYPE_PLMN The contact type is for phone number like +1012345678

6.11.4.2 enum_MSG_ENCODE_TYPE_E

Represents the values of String encoding types.

Enumerator:

MSG_ENCODE_UTF8 The string encoding type is UTF8

MSG_ENCODE_UTF16 The string encoding type is UTF16

MSG_ENCODE_ASCII The string encoding type is ASCII

MSG_ENCODE_GSM7BIT The string encoding type is GSM7BIT

MSG_ENCODE_UCS2 The string encoding type is UCS2

MSG_ENCODE_SHIFT_JIS The string encoding type is Shift JIS

6.11.4.3 enum_MSG_ERROR_E

Represents the values of error codes.

Success code is zero, all error codes SHOULD be negative and smaller than [MSG_ERROR_BEGIN](#)

Enumerator:

MSG_SUCCESS Successful

MSG_ERR_FS_ERROR File system error

MSG_ERR_FS_FULL File system is full

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_INVALID_MSG_TYPE Message type is invalid

MSG_ERR_INVALID_SORT_TYPE Sort type is invalid

MSG_ERR_INVALID_FOLDER_ID Folder Id is invalid

MSG_ERR_INVALID_STORAGE_REG Storage registry is invalid

MSG_ERR_INVALID_CONNECTION Connection is invalid

MSG_ERR_NULL_POINTER Pointer is NULL

MSG_ERR_INVALID_MESSAGE_ID Message id is invalid

MSG_ERR_INVALID_ACCOUNT_ID Account id is invalid

MSG_ERR_INVALID_QUERY_ID Query id is invalid

MSG_ERR_SESSION_NOT_CONNECTED Session is not connected

MSG_ERR_INVALID_SESSION_HANDLE Session handle is invalid

MSG_ERR_PERMISSION_DENIED Permission is denied

MSG_ERR_STORAGE_NOT_INIT Storage is not initialized

MSG_ERR_STORAGE_REG_NOT_INIT Storage registry is not initialized

MSG_ERR_INVALID_SESSION_PRIORITY Session priority is invalid

MSG_ERR_INVALID_FILTER_ID Filter Id is invalid

MSG_ERR_INVALID_REQUEST Request is invalid: Request type is unsupported; The message type in request is invalid.

MSG_ERR_INVALID_FOLDER_TYPE Folder type is invalid

MSG_ERR_INVALID_FOLDER Folder information is invalid

MSG_ERR_NOT_EMPTY_FOLDER Folder is not empty

MSG_ERR_INVALID_ACCOUNT Account information is invalid

MSG_ERR_CONNECTION_NOT_CONNECT Connection is not connected

MSG_ERR_INVALID_MESSAGE Connection is not connected

MSG_ERR_STORAGE_FULL Storage is full

MSG_ERR_ROOTFOLDER_INUSE Root folder is associated with account(s)

MSG_ERR_STORAGE_ERROR Storage is error

MSG_ERR_MEMORY_ERROR Memory is error

MSG_ERR_COMMUNICATION_ERROR Communication between client and server is error

MSG_ERR_MAX_NUMBER_REACHED Max number reached when adding message,account or folder

MSG_ERR_INVALID_REQUEST_ID Request Id is invalid

MSG_ERR_UNKNOWN Unknown errors

MSG_ERR_FRAMEWORK_RESERVE_BEGIN Framework reserved error code begin value

MSG_ERR_FRAMEWORK_RESERVE_END Framework reserved error code end value

MSG_ERR_PLUGIN Generic error code for plugin

MSG_ERR_PLUGIN_DEF_BEGIN_VALUE Plugin defined error code begin value. It's equal to MSG_ERR_PLUGIN

6.11.4.4 enum _MSG_FOLDER_TYPE_E

Represents the values of folder types.

Enumerator:

MSG_FOLDER_TYPE_ROOT Root folder

MSG_FOLDER_TYPE_INBOX InBox folder

MSG_FOLDER_TYPE_OUTBOX OutBox folder

MSG_FOLDER_TYPE_DRAFTS Draft folder

MSG_FOLDER_TYPE_SENT Sent folder

MSG_FOLDER_TYPE_DELETED Trash folder

MSG_FOLDER_TYPE_USERDEFINE Folder is created by user

6.11.4.5 enum _MSG_MAIN_TYPE_E

Represents the values of message main types.

Four message main types are predefined: SMS, MMS, Email and iMode. More main message types can be defined here.

Enumerator:

MSG_MESSAGE_TYPE_UNKNOWN Unknown message type
MSG_MESSAGE_TYPE_MMS MMS message
MSG_MESSAGE_TYPE_EMAIL Email message
MSG_MESSAGE_TYPE_SMS SMS message
MSG_MESSAGE_TYPE_IMODE i-mode message

6.11.4.6 enum _MSG_MEDIA_TYPE_E

Represents the values of media types.

Enumerator:

MSG_MEDIA_FLASH Media type is flash
MSG_MEDIA_REMOVABLE Media type is removable
MSG_MEDIA_SIM Media type is SIM card

6.11.4.7 enum _MSG_MESSAGE_PRIORITY_E

Represents the values of message priority.

Enumerator:

MSG_MESSAGE_PRIORITY_LOW Low priority
MSG_MESSAGE_PRIORITY_NORMAL Normal priority
MSG_MESSAGE_PRIORITY_HIGH High priority

6.11.4.8 enum _MSG_REQUEST_TYPE_E

Represents the values of request types.

This enum is used as the value of type [MSG_REQUEST_TYPE_T](#). Each message plugin can define the supported request types in this enum. Application can use the request type to invoke plugin corresponding operation via framework. Note that the enum between MSG_REQUEST_TYPE_RESERVED_BEGIN and MSG_REQUEST_TYPE_RESERVED_END are reserved for framework. For details of request, please refer to “Request” in *Messaging API* [2] .

Enumerator:

MSG_REQUEST_TYPE_INVALID It's used as the initial value to request type.
MSG_REQUEST_TYPE_RESERVED_BEGIN Framework reserved begin value
MSG_REQUEST_TYPE_SEND Request to send message.

This request is used to send a message. After this request is submitted, following status may be triggered:

MSG_STATUS_SENDING: Message is in sending.

MSG_STATUS_SENT: Message is sent successfully.

MSG_STATUS_FAIL_TO_SEND: Message is fail to send.

If the request is cancelled successfully, the message is not sent and following status is triggered:

MSG_STATUS_REQ_CANCELLED: Request is cancelled.

MSG_REQUEST_TYPE_SENDSAVE Request to send and save message.

This request is used to send and save a message. After this request is submitted, following status may be triggered:

MSG_STATUS_SENDING: Message is in sending.

MSG_STATUS_SAVED_NOTSENT: Message is saved but not sent.

MSG_STATUS_SENT_NOTSAVED: Message is sent but not saved.

MSG_STATUS_SENT_SAVED: Message is both saved and sent.

MSG_STATUS_FAIL_TO_SENDSAVE: Message is neither saved nor sent.

If the request is cancelled successfully, the message is not sent and not saved. Following status is triggered:

MSG_STATUS_REQ_CANCELLED: Request is cancelled.

MSG_REQUEST_TYPE_RESERVED_END Framework reserved end value

MSG_REQUEST_TYPE_USER_DEF_BEGIN User defined request type begin value

6.11.4.9 enum _MSG_SESSION_PRIORITY_E

Represents the values of session priority.

Enumerator:

MSG_SESSION_PRIORITY_LOWEST Lowest priority

MSG_SESSION_PRIORITY_LOW Low priority

MSG_SESSION_PRIORITY_NORMAL Normal priority

MSG_SESSION_PRIORITY_HIGH High priority

MSG_SESSION_PRIORITY_HIGHEST Highest priority

6.11.4.10 enum _MSG_SIZE_UNIT_E

Enumerator:

MSG_SIZE_UNIT_BYTE Unit is byte

MSG_SIZE_UNIT_KBYTE Unit is kilo-byte

MSG_SIZE_UNIT_MBYTE Unit is mega-byte

MSG_SIZE_UNIT_GBYTE Unit is giga-byte

MSG_SIZE_UNIT_NUMBER Unit is number

6.11.4.11 enum _MSG_STATUS_E

Represents the values of status.

This enum is used as the value of type [MSG_STATUS_T](#). The specific status for each message can be defined in this enum. For instance of Email, sending status can be defined to track the email sending progress.

Enumerator:

MSG_STATUS_UNKNOWN Unknown status

MSG_STATUS_READ Message is read
MSG_STATUS_UNREAD Message is unread
MSG_STATUS_REPLIED Message is replied
MSG_STATUS_FORWARDED Message is forwarded
MSG_STATUS_DRAFT Message is draft
MSG_STATUS_EDITING Message is in editing
MSG_STATUS_PROTECTED Message is protected
MSG_STATUS_SENDING Message is sending
MSG_STATUS_SENT Message is sent
MSG_STATUS_FAIL_TO_SEND Message is fail to send
MSG_STATUS_SAVED_NOTSENT Message is saved, but not sent
MSG_STATUS_SENT_NOTSAVED Message is sent, but not saved
MSG_STATUS_SENT_SAVED Message is sent and saved
MSG_STATUS_FAIL_TO_SENDSAVE Message is not saved and not sent
MSG_STATUS_REQ_CANCELLED Request is cancelled
MSG_STATUS_USER_DEF_BEGIN User defined status begin value

6.11.4.12 enum _MSG_STORAGE_ID_VALUE_E

Enumerator:

MSG_STORAGE_UNKNOWN
MSG_STORAGE_DEFAULT The default storage

6.12 Msg_Plugin_Filter_Interface.h File Reference

6.12.1 Detailed Description

Defines the filter interfaces that filter plugin MUST implement.

```
#include "Msg_Mapi_Types.h"
#include "Msg_Mapi_Message.h"
#include "Msg_Mapi_FilterRule.h"
```

Data Structures

- struct [_MSG_PLUGIN_FILTER_MANAGER_S](#)
Represents a filter manager.

Typedefs

- typedef [MSG_ERROR_T](#)(* [MsgPlgFilterInitialize](#))()
The prototype for initializing function of filter manager.
- typedef void(* [MsgPlgFilterFinalize](#))()
The prototype for finalizing function of filter manager.
- typedef bool(* [MsgPlgFilterCheckRules](#))(const [MSG_MESSAGE_S](#) *pMsg, const [MSG_FILTER_S](#) *pFilter, bool *autoStore, bool *intercept)
The prototype for checking filter rule function of filter manager.
- typedef struct [_MSG_PLUGIN_FILTER_MANAGER_S](#) [MSG_PLUGIN_FILTER_MANAGER_S](#)

Functions

- [MSG_ERROR_T](#) [MsgPlgFilterCreateHandle](#) ([MSG_PLUGIN_FILTER_MANAGER_S](#) *pFilterHandle)
Creates a filter manager handle.
- [MSG_ERROR_T](#) [MsgPlgFilterDestroyHandle](#) ()
Destroys a filter manager handle.

6.12.2 Typedef Documentation

6.12.2.1 typedef struct [_MSG_PLUGIN_FILTER_MANAGER_S](#) [MSG_PLUGIN_FILTER_MANAGER_S](#)

6.12.2.2 typedef bool(* [MsgPlgFilterCheckRules](#))(const [MSG_MESSAGE_S](#) *pMsg, const [MSG_FILTER_S](#) *pFilter, bool *autoStore, bool *intercept)

The prototype for checking filter rule function of filter manager.

This function is to check filter rule.

This function check wheather the message satisfies the filter rule, return the `autoStore` and `intercept` properties of current filter rule.

For details, please refer to [Filter Plug-in: Set Filter List to Filter Messages](#).

Parameters:

- ← *pMsg* is the pointer to the message that needs to be filtered.
- ← *pFilter* is the pointer to the filter rule.
- *autoStore* indicates whether the message should be stored by framework automatically. This field is valid only when this function returns `true`.
- *intercept* indicates whether the message should be intercepted. This field is valid only when this function returns `true`.

Returns:

`bool`

Return values:

- true* the message satisfies the filter rule.
- false* the message does not satisfy the filter rule.

6.12.2.3 typedef void(* MsgPlgFilterFinalize)()

The prototype for finalizing function of filter manager.

This function is to do finalization tasks such as releasing memory and releasing file descriptors.

Returns:

`void`

6.12.2.4 typedef MSG_ERROR_T(* MsgPlgFilterInitialize)()

The prototype for initializing function of filter manager.

This function is to do initialization tasks such as allocating memory and opening file descriptors.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* filter plugin initializes successfully.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code ([MSG_PLUGIN_ERROR_T](#)), which will be returned to application when `MsgGetSessionPluginError()` is called. *n* could be 0 if plugin doesn't want to specify the error.

6.12.3 Function Documentation

6.12.3.1 `MSG_ERROR_T MsgPlgFilterCreateHandle (MSG_PLUGIN_FILTER_MANAGER_S *pFilterHandle)`

Creates a filter manager handle.

This function is to create the filter manager handle. This is the only way to get the filter manager handle.

Parameters:

→ *pFilterHandle* is a pointer to the filter manager handle. When this function return `MSG_SUCCESS`, it will point to a valid filter manager handle. Otherwise it will be set to NULL.

Returns:

`MSG_ERROR_T`

Return values:

MSG_SUCCESS creates the filter plugin handle successfully.

MSG_ERR_NULL_POINTER *pFilterHandle* is NULL.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetSessionPluginError()` is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.12.3.2 `MSG_ERROR_T MsgPlgFilterDestroyHandle ()`

Destroys a filter manager handle.

This function is to destroy the filter manager handle. This is the only way to destroy the filter manager handle. After destroying, the handle can no longer be used.

Returns:

`MSG_ERROR_T`

Return values:

MSG_SUCCESS destroys the filter plugin handle successfully.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetSessionPluginError()` is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13 Msg_Plugin_Storage_Interface.h File Reference

6.13.1 Detailed Description

Defines the interfaces that message storage plugin MUST implement.

These functions can be divided into three categories:

- Storage management.
- Message storage.
- Account storage.

```
#include "Msg_Mapi_Types.h"
#include "Msg_Mapi_Storage_Manager.h"
#include "Msg_Plugin_Types.h"
#include "Msg_Mapi_Message.h"
#include "Msg_Mapi_Account.h"
#include "Msg_Framework_Storage_Interface.h"
```

Data Structures

- struct [_MSG_PLG_STORAGE_S](#)
Represents a message storage plugin handle.

Typedefs

- typedef [MSG_ERROR_T](#)(* [MsgPlgStoSetFwStoHandle](#))(MSG_FW_STORAGE_S *pFwStoHandle)
The prototype for setting framework storage plugin handle.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoAttachStorage](#))(MSG_STORAGE_ID_T StorageId)
The prototype for attaching a specific storage by storage id.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoDetachStorage](#))(MSG_STORAGE_ID_T StorageId)
The prototype for detaching a specific storage by storage id..
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoCheckStatus](#))(MSG_STORAGE_ID_T StorageId, [MSG_PLG_STORAGE_STATUS_T](#) *pStatus)
The prototype for checking the status of the storage by storage id.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoInitStorage](#))(MSG_STORAGE_ID_T StorageId)
The prototype for initializing the storage.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoGetStorageStatus](#))(MSG_STORAGE_ID_T storageId, unsigned long long *totalSize, unsigned long long *usedSize, [MSG_SIZE_UNIT_T](#) *unit)
Gets specific message storage status by storage id.

- typedef [MSG_ERROR_T](#)(* [MsgPlgStoAddMessage](#))(MSG_MESSAGE_S *pMsg)
The prototype for adding a message into storage.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoUpdateMessage](#))(const MSG_MESSAGE_S *pMsg)
The prototype for updating a Message.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoDeleteMessage](#))(MSG_MESSAGE_ID_T MsgId)
The prototype for deleting a message.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoDelAllMsgInFolder](#))(MSG_FOLDER_ID_T FolderId)
Deletes all messages in the specified folder.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoGetMessage](#))(MSG_MESSAGE_ID_T MsgId, MSG_MESSAGE_S **ppMsg)
The prototype for getting the message detail information.
- typedef void(* [MsgPlgStoReleaseMessage](#))(MSG_MESSAGE_S *pMsg)
The prototype for releasing the memory of MSG_MESSAGE_S allocated in [MsgPlgStoGetMessage](#).
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoAddAccount](#))(MSG_ACCOUNT_S *pAccount)
The prototype for adding a new account in message storage plugin.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoUpdateAccount](#))(const MSG_ACCOUNT_S *pAccount)
The prototype for updating an account.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoDeleteAccount](#))(MSG_ACCOUNT_ID_T accountId)
The prototype for deleting an account.
- typedef [MSG_ERROR_T](#)(* [MsgPlgStoGetAccount](#))(MSG_ACCOUNT_ID_T accountId, MSG_ACCOUNT_S **ppAccount)
The prototype for getting the detailed account information for the specified account.
- typedef void(* [MsgPlgStoReleaseAccount](#))(MSG_ACCOUNT_S *pAccount)
The prototype for releasing the memory block of a account structure.
- typedef struct [_MSG_PLG_STORAGE_S](#) MSG_PLG_STORAGE_S

Functions

- [MSG_ERROR_T](#) [MsgPlgStoCreateHandle](#) (MSG_PLG_STORAGE_S *hPLGStorageHandle)
Creates a message storage plugin handle.
- [MSG_ERROR_T](#) [MsgPlgStoDestroyHandle](#) ()
Destroys a message storage plugin handle.

6.13.2 Typedef Documentation

6.13.2.1 typedef struct _MSG_PLG_STORAGE_S MSG_PLG_STORAGE_S

6.13.2.2 typedef MSG_ERROR_T(* MsgPlgStoAddAccount)(MSG_ACCOUNT_S *pAccount)

The prototype for adding a new account in message storage plugin.

This function is to add a new account into message storage plugin. It MUST call the framework plugin's corresponding function [MsgFwStoAddAccount](#) to synchronize account common information with framework storage. This function MUST guarantee the data integrity between framework storage plugin and message storage plugin. Mandatory fields of account structure MUST be valid, otherwise the function MUST be failed. The mandatory fields for adding operation are `accountId`, `msgType`, `rootFolderId`, `storageId`, `defaultMsgStorageId`.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#).

Parameters:

← *pAccount* contains the data user defined for this new account.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER `pAccount` is NULL.

MSG_ERR_INVALID_ACCOUNT Mandatory fields of account structure are invalid.

MSG_ERR_STORAGE_FULL Storage is FULL.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

`n` is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. `n` could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.3 typedef MSG_ERROR_T(* MsgPlgStoAddMessage)(MSG_MESSAGE_S *pMsg)

The prototype for adding a message into storage.

This function is to add a message into message plugin storage. It should call framework storage plugin's corresponding [MsgFwStoAddMessage](#) to synchronize common message information with framework storage. This method should guarantee the data integrity between framework storage and message plugin storage. Mandatory fields of message structure MUST be valid, otherwise the function will be failed. The mandatory fields for adding operation are `msgId`, `parentId`, `msgType`, `storageId`.

If `storageId` is `MSG_STORAGE_DEFAULT`, the message MUST be added to the `defaultMsgStorageId` in its associated account.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *pMsg* is a pointer to a `MSG_MESSAGE_S` struct.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER `pMsg` is NULL.

MSG_ERR_INVALID_MESSAGE Mandatory fields of message are invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_STORAGE_FULL Storage is FULL.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

`n` is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetStoragePluginError()` is called. `n` could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.4 `typedef MSG_ERROR_T(* MsgPlgStoAttachStorage)(MSG_STORAGE_ID_T StorageId)`

The prototype for attaching a specific storage by storage id.

If the message storage plugin uses the database, it can establish connection with the database and open the table in this function.

If the message storage plugin uses the files, it can open the files in this function.

For storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *StorageId* is an integer value indicating which storage.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.5 typedef MSG_ERROR_T(* MsgPlgStoCheckStatus)(MSG_STORAGE_ID_T StorageId, MSG_PLG_STORAGE_STATUS_T *pStatus)

The prototype for checking the status of the storage by storage id.

Parameters:

← **StorageId** is an integer value indicating which storage.

→ **pStatus** indicates the status of the storage.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_NULL_POINTER pStatus is NULL.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.6 typedef MSG_ERROR_T(* MsgPlgStoDelAllMsgInFolder)(MSG_FOLDER_ID_T FolderId)

Deletes all messages in the specified folder.

This function is to delete all messages in the specified folder, but it will not delete the specified folder's subfolders and the messages in the subfolders.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *FolderId* is a folder Id.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_INVALID_FOLDER_ID Folder id is invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code ([MSG_PLUGIN_ERROR_T](#)), which will be returned to application when `MsgGetStoragePluginError()` is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.7 `typedef MSG_ERROR_T(* MsgPlgStoDeleteAccount)(MSG_ACCOUNT_ID_T accountId)`

The prototype for deleting an account.

This function is to delete an account in message storage plugin, and it MUST call the framework plugin memethod [MsgFwStoDeleteAccount](#) to delete corresponding account common information in framework storage. This method MUST guarantee the data integrity between framework storage and message storage plugin.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#).

Parameters:

← *accountId* identifies the specific account need to be deleted.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_INVALID_ACCOUNT_ID This account does not exist.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code ([MSG_PLUGIN_ERROR_T](#)), which will be returned to application when `MsgGetStoragePluginError()` is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.8 typedef MSG_ERROR_T(* MsgPlgStoDeleteMessage)(MSG_MESSAGE_ID_T MsgId)

The prototype for deleting a message.

This function is to delete a message in message plugin storage. It should call framework storage plugin's corresponding [MsgFwStoDeleteMessage](#) to synchronize common message information with framework storage. This method should guarantee the data integrity between framework storage and message plugin storage.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *MsgId* indicates to be deleted message's Id.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_INVALID_MESSAGE_ID Message id is invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.9 typedef MSG_ERROR_T(* MsgPlgStoDetachStorage)(MSG_STORAGE_ID_T StorageId)

The prototype for detaching a specific storage by storage id..

If the message storage plugin uses the database, it can close connection with the database in this function.

If the message storage plugin uses the files, it can close the files in this function.

For storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *StorageId* is an integer value indicating which storage.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_PLUGIN+*n* the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.10 `typedef MSG_ERROR_T(* MsgPlgStoGetAccount)(MSG_ACCOUNT_ID_T accountId, MSG_ACCOUNT_S **ppAccount)`

The prototype for getting the detailed account information for the specified account.

This function is to get the detailed account information for the specified account. The memory of out parameter `ppAccount` MUST be allocated in this function. For this account may has its own extended structure, The function's caller MUST release its memory by invoking [MsgPlgStoReleaseAccount](#). However if this function is failed, the memory is NOT allocated in this function.

Parameters:

← **accountId** indicates the queried account.

→ **ppAccount** is a pointer to a pointer to the structure which contains account information.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER `ppAccount` is NULL.

MSG_ERR_INVALID_ACCOUNT_ID This account does not exist.

MSG_ERR_PLUGIN+*n* the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.11 `typedef MSG_ERROR_T(* MsgPlgStoGetMessage)(MSG_MESSAGE_ID_T MsgId, MSG_MESSAGE_S **ppMsg)`

The prototype for getting the message detail information.

This function is to return the message detail information, including both common extension data (`pCommExtData`) and specific message extension data (`pExtData`).

This function should call [MsgFwStoGetMsgCommInfo](#) provided by framework storage plugin to get the message common information. The memory for the common information is allocated in [MsgFwStoGetMsgCommInfo](#). This function should allocate the memory for the specific extension data (`pExtData`). However if this function is failed, the memory is NOT allocated in this function.

Parameters:

- ← *MsgId* indicates to be returned message's Id.
- *ppMsg* is a pointer to the message pointer.

Returns:

[MSG_ERROR_T](#)

Return values:

- MSG_SUCCESS* Succeeds in operation.
- MSG_ERR_INVALID_MESSAGE_ID* Message id is invalid.
- MSG_ERR_NULL_POINTER* *ppMsg* is NULL.
- MSG_ERR_STORAGE_ERROR* Storage error.
- MSG_ERR_PLUGIN+n* the error code specific for plugin.

Note:

n is the plugin specific error code ([MSG_PLUGIN_ERROR_T](#)), which will be returned to application when `MsgGetStoragePluginError()` is called. *n* could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.12 `typedef MSG_ERROR_T(* MsgPlgStoGetStorageStatus)(MSG_STORAGE_ID_T storageId, unsigned long long *totalSize, unsigned long long *usedSize, MSG_SIZE_UNIT_T *unit)`

Gets specific message storage status by storage id.

This function is to get total size and used size of the specified message storage.

Parameters:

- ← *storageId* is an integer value indicating which storage.
- *totalSize* the total size in this storage.
- *usedSize* used size in this storage.
- *unit* the unit of the size.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER totalSize is NULL or usedSize is NULL or unit is NULL.

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_COMMUNICATION_ERROR communication between client and server is error.

MSG_ERR_MEMORY_ERROR memory is error.

MSG_ERR_PLUGIN the generic error code for plugin. Call MsgGetStoragePluginError() to get detail error code.

Since:

1.0

6.13.2.13 typedef MSG_ERROR_T(* MsgPlgStoInitStorage)(MSG_STORAGE_ID_T StorageId)

The prototype for initializing the storage.

This function is to create all the data for message storage plugin on the storage.

If the plugin uses the database, it can create tables in the database in this function.

If the plugin uses the files, it can create the files in this function.

When this function returns successfully, all the data on this storage will be filled with default values. If the storage data already exists, this function should reset all the values.

Parameters:

← **StorageId** is an integer value indicating which storage.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation, data on this storage will be reseted.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_INVALID_STORAGE Storage id is invalid

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.14 `typedef void(* MsgPlgStoReleaseAccount)(MSG_ACCOUNT_S *pAccount)`

The prototype for releasing the memory block of a account structure.

This function is to release the memory of a account structure. Message storage plugin MUST implement this function to release the memory of its own extended structure, and in this function it MUST call the framework storage plugin's corresponding function [MsgFwStoReleaseCommAccount](#) to release common extended structure explicitly. This account memory is created by function [MsgPlgStoGetAccount](#).

If pAccount is NULL, nothing happens. If pAccount is invalid, undefined behavior.

Parameters:

← *pAccount* is a pointer to a account structure.

Since:

1.0

6.13.2.15 `typedef void(* MsgPlgStoReleaseMessage)(MSG_MESSAGE_S *pMsg)`

The prototype for releasing the memory of MSG_MESSAGE_S allocated in [MsgPlgStoGetMessage](#).

This function is to release the memory for the specific extension data (pExtData) and it should also call [MsgFwStoReleaseMsgCommon](#) to release the message common information allocated in framework storage plugin.

If pMsg is NULL, nothing happens. If pMsg is invalid, undefined behavior.

Parameters:

← *pMsg* is a pointer to an MSG_MESSAGE_S structure.

Since:

1.0

6.13.2.16 `typedef MSG_ERROR_T(* MsgPlgStoSetFwStoHandle)(MSG_FW_STORAGE_S *pFwStoHandle)`

The prototype for setting framework storage plugin handle.

This function is to set the framework storage plugin handle to message storage plugin. The framework storage plugin handle is used to invoke the functions provided by framework storage plugin. This function will be called when the message storage plugin is firstly lauched by messaging framework. When the framework storage plugin is closed by messaging framework, this method will be called with parameter NULL.

Parameters:

← *pFwStoHandle* the pointer to framework storage plugin handle.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.17 typedef MSG_ERROR_T(* MsgPlgStoUpdateAccount)(const MSG_ACCOUNT_S *pAccount)

The prototype for updating an account.

The function is expected to update msgType, rootFolderId, storageId and defaultMsgStorageID for the account indentified by the given accountId as long as the given values are valid. Optionally, the function updates extension data given by pExtData and extDataSize, and accountName, if necessary.

This function MUST call the framework plugin's corresponding function [MsgFwStoUpdateAccount](#) to update common account information in framework storage. All account information except for account id can be updated. This method MUST guarantee the data integrity between framework storage and message storage plugin.

If pAccount->storageId is different than original one, the account MUST be moved to updated storage.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#).

Parameters:

← **pAccount** contains the data user defined to update account.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER pAccount is NULL.

MSG_ERR_INVALID_ACCOUNT Mandatory fields of account structure are invalid.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.2.18 typedef MSG_ERROR_T(* MsgPlgStoUpdateMessage)(const MSG_MESSAGE_S *pMsg)

The prototype for updating a Message.

The function is to update message data for the message indentified by the given msgId as long as the given values are valid in message plugin storage.

It MUST call framework storage plugin's corresponding [MsgFwStoUpdateMessage](#) to synchronize common message information with framework storage. This method MUST guarantee the data integrity between framework storage and message plugin storage. Following fields of message structure MUST be valid, otherwise the function will be failed:

msgId, parentId, msgType, storageId.

If pMsg->parentId is different than original one, the message MUST be moved to updated destination folder.

If pMsg->storageId is different than original one, the message MUST be moved to updated storage.

For details about storage plugin mechanism, please refer to [Framework Storage Plug-in](#) and [Message Storage Plug-in](#)

Parameters:

← *pMsg* is a const pointer to a MSG_MESSAGE_S structure.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER pMsg is NULL.

MSG_ERR_INVALID_MESSAGE Mandatory fields of message are invalid.

MSG_ERR_STORAGE_ERROR Storage error.

MSG_ERR_MAX_NUMBER_REACHED max number reached.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetStoragePluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.13.3 Function Documentation

6.13.3.1 MSG_ERROR_T MsgPlgStoCreateHandle (MSG_PLG_STORAGE_S *hPLGStorageHandle)

Creates a message storage plugin handle.

This function is to provide message storage plugin handle. This is the only way to get the message storage plugin handle.

Parameters:

→ *hPLGStorageHandle* is the pointer to a message storage plugin handle.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_NULL_POINTER *hPLGStorageHandle* is NULL.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (*MSG_PLUGIN_ERROR_T*), which will be returned to application when *MsgGetSessionPluginError()* is called. *n* could be 0 if plugin doesn't want to specify the error

Since:

1.0

6.13.3.2 MSG_ERROR_T MsgPlgStoDestroyHandle ()

Destroys a message storage plugin handle.

This function is to destroy a message storage plugin handle. After destroying the handle, the handle can not be used anymore.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS Succeeds in operation.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (*MSG_PLUGIN_ERROR_T*), which will be returned to application when *MsgGetSessionPluginError()* is called. *n* could be 0 if plugin doesn't want to specify the error

Since:

1.0

6.14 Msg_Plugin_Transport_Interface.h File Reference

6.14.1 Detailed Description

Defines the transport interfaces that transport plugin MUST implement.

```
#include "Msg_Mapi_Types.h"
#include "Msg_Mapi_Message.h"
#include "Msg_Mapi_Message_Status.h"
#include "Msg_Mapi_FilterRule.h"
#include "Msg_Mapi_Message_Request.h"
#include "Msg_Mapi_Account.h"
```

Data Structures

- struct [_MSG_TRANSPORT_LISTENER_S](#)
Represents a transport listener handle.
- struct [_MSG_PLUGIN_TRANSPORT_S](#)
Represents a transport plugin handle.

Typedefs

- typedef struct [_MSG_PLUGIN_TRANSPORT_S](#) [MSG_PLUGIN_TRANSPORT_S](#)
- typedef [MSG_ERROR_T](#)(* [MsgPlgTransOnStatusChanged](#))(MSG_PLUGIN_TRANSPORT_S *pTransHandle, [MSG_MESSAGE_STATUS_S](#) *pStatus)
The prototype for onStatusChanged callback of framework.
- typedef [MSG_ERROR_T](#)(* [MsgPlgTransOnMsgIncoming](#))(MSG_PLUGIN_TRANSPORT_S *pTransHandle, [MSG_MESSAGE_S](#) *pMsg, [MSG_MESSAGE_STATUS_S](#) *pStatus, int *uid)
The prototype for onMsgIncoming callback of framework.
- typedef struct [_MSG_TRANSPORT_LISTENER_S](#) [MSG_TRANSPORT_LISTENER_S](#)
- typedef [MSG_ERROR_T](#)(* [MsgPlgTransInitialize](#))()
The prototype for initializing function of transport plugin.
- typedef void(* [MsgPlgTransFinalize](#))()
The prototype for finalizing function of transport plugin.
- typedef [MSG_ERROR_T](#)(* [MsgPlgTransSubmitRequest](#))(const [MSG_MESSAGE_REQUEST_S](#) *pRequest, const [MSG_ACCOUNT_S](#) *pAccount)
The prototype for submitting request function of transport plugin.
- typedef [MSG_ERROR_T](#)(* [MsgPlgTransCancelRequest](#))(int reqId)
The prototype for cancelling request function of transport plugin.

- typedef `MSG_ERROR_T(* MsgPlgTransRegisterListener )(MSG_TRANSPORT_LISTENER_S *pListener)`

The prototype for registering listener function of transport plugin.

- typedef `MSG_ERROR_T(* MsgPlgTransDiscardMsg )(int uid)`

The prototype for discarding an incoming message.

Functions

- `MSG_ERROR_T MsgPlgTransCreateHandle (MSG_PLUGIN_TRANSPORT_S *pTransHandle)`

Creates a transport plugin handle.

- `MSG_ERROR_T MsgPlgTransDestroyHandle ()`

Destroys a transport plugin handle.

6.14.2 Typedef Documentation

6.14.2.1 typedef struct `_MSG_PLUGIN_TRANSPORT_S` `MSG_PLUGIN_TRANSPORT_S`

6.14.2.2 typedef struct `_MSG_TRANSPORT_LISTENER_S` `MSG_TRANSPORT_LISTENER_S`

6.14.2.3 typedef `MSG_ERROR_T(* MsgPlgTransCancelRequest)(int reqId)`

The prototype for cancelling request function of transport plugin.

This function is to cancel the request from framework according to the request Id.

This is an asynchronous function.

Parameters:

← *reqId* is the request Id that refers to the request.

Returns:

`MSG_ERROR_T`

Return values:

`MSG_SUCCESS` cancels the request successfully.

`MSG_ERR_PLUGIN_BUSY` transport plugin is busy to handle the request. In this case, framework will pass this error code to application.

`MSG_ERR_PLUGIN+n` the error code specific for plugin.

Note:

n is the plugin specific error code (`MSG_PLUGIN_ERROR_T`), which will be returned to application when `MsgGetSessionPluginError()` is called. n could be 0 if plugin doesn't want to specify the error.

6.14.2.4 typedef MSG_ERROR_T(* MsgPlgTransDiscardMsg)(int uid)

The prototype for discarding an incoming message.

This function is to discard a message. If no application is interested in the message, the framework invoke this function to inform plugin to discard the message.

Parameters:

← *uid* is the unique Id of the message.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS discards the message successfully.

MSG_ERR_INVALID_MESSAGE_UID the uid of the message is invalid.

MSG_ERR_PLUGIN_BUSY transport plugin is busy to handle the request. In this case, framework will pass this error code to application.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetSessionPluginError() is called. n could be 0 if plugin doesn't want to specify the error.

6.14.2.5 typedef void(* MsgPlgTransFinalize)()

The prototype for finalizing function of transport plugin.

This function is to do such finalization works as deregistering network etc.

Returns:

void

6.14.2.6 typedef MSG_ERROR_T(* MsgPlgTransInitialize)()

The prototype for initializing function of transport plugin.

This function is to do such initialization works as registering network etc.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS transport plugin initializes successfully.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetSessionPluginError() is called. n could be 0 if plugin doesn't want to specify the error

6.14.2.7 `typedef MSG_ERROR_T(* MsgPlgTransOnMsgIncoming)(MSG_PLUGIN_TRANSPORT_S *pTransHandle, MSG_MESSAGE_S *pMsg, MSG_MESSAGE_STATUS_S *pStatus, int *uid)`

The prototype for onMsgIncoming callback of framework.

This callback function will be invoked when a new message is incoming.

Transport plugin MUST NOT implement this callback. Framework implement this callback function. Transport plugin just informs the framework the new message incoming event by calling this callback function. pMsg and pStatus will be transferred to application.

Transport plugin allocates the memory for pMsg and pStatus. After MSG_PLG_Trans_onReceiveMessage callback function returns, transport plugin should release the memory.

Parameters:

- ← *pTransHandle* is the pointer to the transport plugin handle.
- ← *pMsg* is the pointer to the incoming message. Plugin should allocate the memory and release it when this callback function returns.
- ← *pStatus* The status of the message. Plugin should allocate the memory and release it when this callback function returns.
- ← *uid* is the unique id of the incoming message. Framework should keep this id, if no application is interested in the message, this id should be passed back to plugin to discard the message.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS transport plugin informs the framework the new message incoming event successfully.

MSG_ERR_INVALID_TRANS_HANDLE the transport plugin handle is invalid.

MSG_ERR_NULL_POINTER pMsg is NULL or pStatus is NULL or uid is NULL.

6.14.2.8 `typedef MSG_ERROR_T(* MsgPlgTransOnStatusChanged)(MSG_PLUGIN_TRANSPORT_S *pTransHandle, MSG_MESSAGE_STATUS_S *pStatus)`

The prototype for onStatusChanged callback of framework.

This callback function will be invoked when a message request's status changes. Transport plugin MUST NOT implement this callback. Framework implements this callback function. Transport plugin just informs the framework the statusChanged event by calling this callback function. pStatus will be transferred to application.

Transport plugin allocates the memory for pStatus. After MSG_PLG_Trans_onStatusChanged callback function returns, transport plugin should release the memory.

Parameters:

- ← *pTransHandle* is the pointer to the transport plugin handle.
- ← *pStatus* is the pointer to the changed status. Plugin shall allocate the memory and release it after call this callback function.

Returns:[MSG_ERROR_T](#)**Return values:***MSG_SUCCESS* transport plugin informs the framework the status changed event successfully.*MSG_ERR_NULL_POINTER* pStatus is NULL.*MSG_ERR_INVALID_TRANS_HANDLE* the transport plugin handle is invalid.**6.14.2.9 typedef MSG_ERROR_T(* MsgPlgTransRegisterListener)(MSG_TRANSPORT_LISTENER_S *pListener)**

The prototype for registering listener function of transport plugin.

This function is to register transport listener. When transport event occurs (status change/new message incomes), callback function in transport listener should be invoked to inform the listener.

Parameters:

← *pListener* is the pointer to the transport listener that need to be registered into the transport plugin.
If NULL is passed in, the previous registered listener will be deregistered.

Returns:[MSG_ERROR_T](#)**Return values:***MSG_SUCCESS* register the listener successfully.*MSG_ERR_NULL_POINTER* pListener is NULL.*MSG_ERR_PLUGIN+n* the error code specific for plugin.**Note:**

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetSessionPluginError() is called. n could be 0 if plugin doesn't want to specify the error.

6.14.2.10 typedef MSG_ERROR_T(* MsgPlgTransSubmitRequest)(const MSG_MESSAGE_REQUEST_S *pRequest, const MSG_ACCOUNT_S *pAccount)

The prototype for submitting request function of transport plugin.

This function is to submit the request from framework.

If pRequest->reqType is invalid, MSG_ERR_INVALID_REQUEST MUST be returned.

If pRequest->pMsg->msgType is not same to pRequest->msgType, MSG_ERR_INVALID_REQUEST MUST be returned.

Following fields MUST be ignored:

pReq->pMsg->msgId, pReq->pMsg->parentId, pReq->pMsg->storageId.

Parameters:

← *pRequest* is the pointer to the request from framework.

← *pAccount* is the pointer to the account from framework.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS submits request successfully.

MSG_ERR_INVALID_REQUEST some information in request are invalid.

MSG_ERR_PLUGIN_BUSY transport plugin is busy to handle the request. In this case, framework will pass this error code to application.

MSG_ERR_NULL_POINTER pRequest is NULL or pAccount is NULL.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetSessionPluginError() is called. n could be 0 if plugin doesn't want to specify the error.

6.14.3 Function Documentation

6.14.3.1 **MSG_ERROR_T** MsgPlgTransCreateHandle (MSG_PLUGIN_TRANSPORT_S * pTransHandle)

Creates a transport plugin handle.

This function is to provide transport plugin handle. This is the only way to get the transport plugin handle.

Parameters:

→ *pTransHandle* is a pointer to transport plugin handle. When this function return [MSG_SUCCESS](#), it will point to a valid transport plugin handle. Otherwise, it will be set to NULL.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS creates the transport plugin handle successfully.

MSG_ERR_NULL_POINTER pTransHandle is NULL.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetSessionPluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.14.3.2 MSG_ERROR_T MsgPlgTransDestroyHandle ()

Destroys a transport plugin handle.

This function is to destroy the transport plugin handle. After destroying the handle, the handle can not be used anymore.

Returns:

[MSG_ERROR_T](#)

Return values:

MSG_SUCCESS destroys the transport plugin handle successfully.

MSG_ERR_PLUGIN+n the error code specific for plugin.

Note:

n is the plugin specific error code (MSG_PLUGIN_ERROR_T), which will be returned to application when MsgGetSessionPluginError() is called. n could be 0 if plugin doesn't want to specify the error.

Since:

1.0

6.15 Msg_Plugin_Types.h File Reference

6.15.1 Detailed Description

Defines general types for plugins.

```
#include "Msg_Mapi_Types.h"
```

Typedefs

- typedef unsigned int [MSG_PLG_STORAGE_STATUS_T](#)

Represents a storage status for plugins.

Enumerations

- enum [_MSG_PLG_STORAGE_STATUS_E](#) { [MSG_PLG_MEDIA_NOT_AVAILABLE](#), [MSG_PLG_DATA_NOT_AVAILABLE](#), [MSG_PLG_DATA_AVAILABLE](#) }

Represents the values of storage status for plugins.

- enum [_MSG_PLG_ERROR_E](#) { [MSG_ERR_INVALID_TRANS_HANDLE](#) = [MSG_ERR_FRAMEWORK_RESERVE_BEGIN](#) - 1, [MSG_ERR_INVALID_MESSAGE_UID](#) = [MSG_ERR_FRAMEWORK_RESERVE_BEGIN](#) - 2, [MSG_ERR_PLUGIN_BUSY](#) = [MSG_ERR_FRAMEWORK_RESERVE_BEGIN](#) - 3 }

Represents the values of error codes for plugins.

- enum [_MSG_SUB_TYPE_E](#) { [MSG_MSG_TYPE_SMS_TEXT](#) = 0 }
- enum [_MSG_FRAMEWORK_ERROR_E](#) { [MSG_ERR_INVALID_PLUGIN_HANDLE](#) = [MSG_ERR_FRAMEWORK_RESERVE_BEGIN](#) - 1 }

6.15.2 Typedef Documentation

6.15.2.1 typedef unsigned int MSG_PLG_STORAGE_STATUS_T

Represents a storage status for plugins.

The values for this type SHOULD be in [_MSG_PLG_STORAGE_STATUS_E](#)

6.15.3 Enumeration Type Documentation

6.15.3.1 enum _MSG_FRAMEWORK_ERROR_E

Enumerator:

MSG_ERR_INVALID_PLUGIN_HANDLE

6.15.3.2 enum _MSG_PLG_ERROR_E

Represents the values of error codes for plugins.

The values of error codes SHOULD be between MSG_ERR_FRAMEWORK_RESERVE_BEGIN and MSG_ERR_FRAMEWORK_RESERVE_END.

Enumerator:

MSG_ERR_INVALID_TRANS_HANDLE Transport handler is invalid

MSG_ERR_INVALID_MESSAGE_UID Message UID is invalid

MSG_ERR_PLUGIN_BUSY Plugin is busy

6.15.3.3 enum _MSG_PLG_STORAGE_STATUS_E

Represents the values of storage status for plugins.

Enumerator:

MSG_PLG_MEDIA_NOT_AVAILABLE Storage media is not available

MSG_PLG_DATA_NOT_AVAILABLE Storage media is ready but no data available

MSG_PLG_DATA_AVAILABLE Storage media is ready and data is available too

6.15.3.4 enum _MSG_SUB_TYPE_E

Enumerator:

MSG_MSG_TYPE_SMS_TEXT

Chapter 7

Page Documentation

7.1 Overview of the LiMo Messaging Plug-ins

7.1.1 Plug-ins Classifying Plug-ins

There are two dimension to classify plug-ins. The first dimension classifies the plug-in according to the following two categories:

- Framework, defined in `framework_plug-in` directory
- Service-specific, defined in `msg_plug-in` directory

A plug-in must belong to one of these categories. The second dimension classifies the plug-in by the modules that the LiMo Messaging Framework provides for applications:

- Filter (only in Framework)
- Transport (only in Framework)
- Storage (in both Framework and Service-specific)

These categories are described in the following sections.

The next figure shows the messaging system and the role of the Messaging Framework, plug-ins and applications in creating and receiving messages.

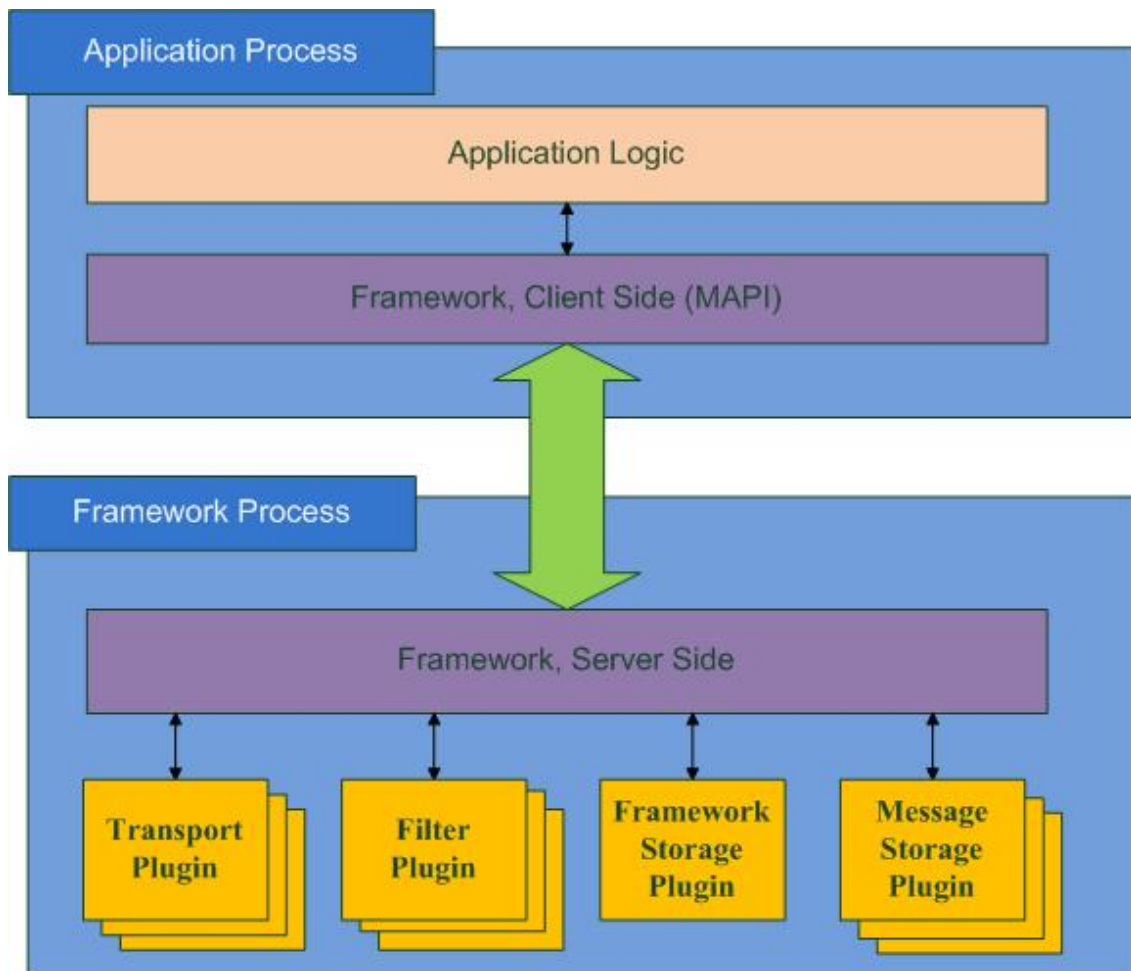


Figure 7.1: Messaging System Architecture

7.1.1.1 Filter Plug-in

There are two groups of functionality that filter plug-ins provide:

- Defining and implementing concrete filter rule. This is used by applications.
- Checking filter rule functions. This is used by the framework to check application's filter rules and decide how to dispatch the incoming messages.

The filter plug-in will be invoked in the framework once MAPI Session-related API is invoked. For more information on MAPI Session-related API, please refer to Message sending and receiving . The definition of this interface is in [Msg_Plugin_Filter_Interface.h](#)

7.1.1.2 Transport Plug-in

The transport plug-in is a bridge between the framework and the network. There are two occasions when this plug-in is used:

- The framework gets the network events from transport plug-in, then notify applications. It is responsible for notifying framework when a network event occurs.
- The framework gets the requests from applications, then sends them to transport plug-in. The transport plug-in is responsible for translating the requests from the framework, then sending them to network (or other components) to get the corresponding service (e.g. send a SMS.)

The transport plug-in will be invoked in the framework once MAPI Session-related API is invoked. For more information on MAPI Session-related API, please refer to Message sending and receiving .

The definition of this interface is in [Msg_Plugin_Transport_Interface.h](#)

7.1.1.3 Framework Storage Plug-in

The framework storage plug-in provides the storage related functionality, such as adding, updating, deleting, querying, counting, moving and getting common information of messages, accounts and folders. This common information is a collection of the generic information from each service-specific message plug-in storage, such as SMS, MMS, Email, and i-Mode messaging. The detail data will be stored in service-specific message storage plug-in. Please refer to Message storage plug-in.

The purpose of creating the common information is for improving performance. The functions, such as adding, updating, deleting, moving data, in framework storage plug-ins will be called by message storage plug-in's corresponding functions to synchronize common information. The other functions of Framework Storage Plug-in, such as querying and sorting, will be called directly by MAPI Storage corresponding methods for performance reasons.

The definition of this interface is in [Msg_Framework_Storage_Interface.h](#).

7.1.1.4 Message Storage Plug-in

Each service-specific message storage plug-in provides the service-specific storage related functionality, such as adding, updating, deleting, moving and getting detail information of messages and accounts. Each service-specific message storage plug-in has its own detail information of message and account.

The purpose of creating the service-specific message detail information is to store the service-specific data.

For example, the SMS storage plug-in has the specific knowledge on SMS format, and additional data that generic message structure does not cover. Each function of the message storage plug-in will be called by the corresponding functions in MAPI storage APIs. The functions in message storage plug-in, such as adding, updating, deleting, and moving data, SHOULD call framework storage plug-in's corresponding methods to synchronize common information.

The definition of this interface is in [Msg_Plugin_Storage_Interface.h](#).

7.1.1.5 Storage Registry

Registry file information has the following format:

```
[Groupname]
mediatype=
storeId=
canDeregister=
name=
path=
```

- Groupname is the group name for this configuration information. It collects the following information to associate it with a specific item of registry.
- mediatype is an integer. It indicates the type of storage media location.
- storeId is a unique storage identifier.
- canDeregister is an integer. Valid values are: 0 for can deregister and 1 for cannot deregister.. Indicates whether or not the storage can be deregistered.
- name is a string. It indicates the name of the storage.
- path is a string. It indicates the full path of the storage location.

The media type 1 for flash and media type 2 for SIM is defined in the following example:

```
[storage-registry-item1]
mediatype=1
storeId=2
canDeregister=0
name= ***
path=/usr/local/...
[storage-registry-item2]
mediatype=2
storeId=1
canDeregister=1
name= ***
path=/usr/local/...
```

There is only one storage registry file. All storage id information shall be written into it. Application developers MUST predefine standard storage ids and write the supported storage information into the storage registry file.

When an application calls `MsgListStorageRegistry` or `MsgGetStorageRegistry` (see *Messaging API* [2]) the messaging framework will receive this command. Then the messaging framework will load and parse this storage registry file and return the result to the application.

7.1.1.6 Storage Subsystem

The following diagram represents the storage subsystem architecture.

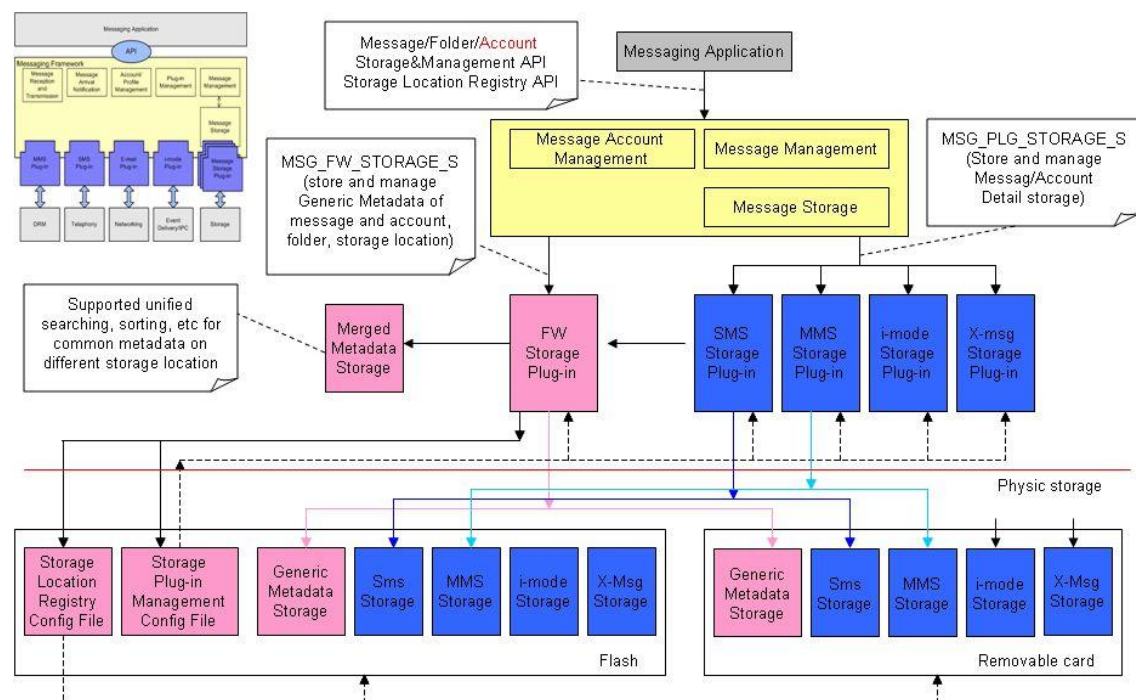


Figure 7.2: Storage Subsystem Architecture Option 1

Alternately, you can combine the framework storage plug-in and the message storage plug-in into one, as the diagram below shows.

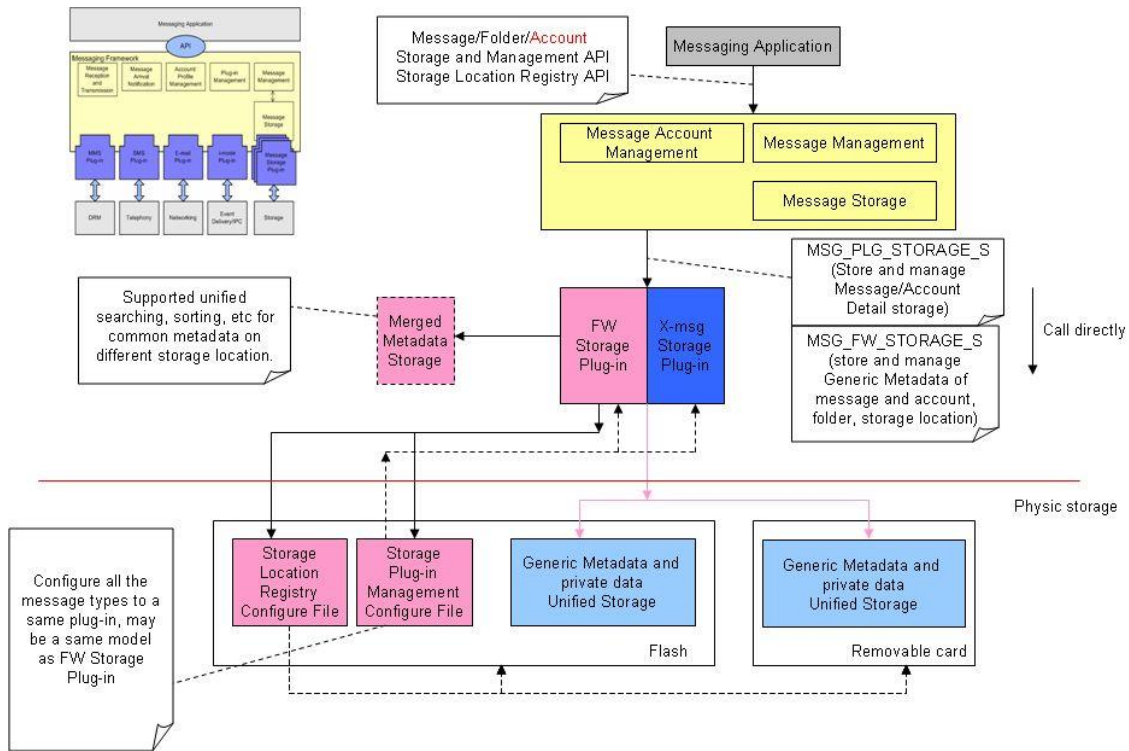


Figure 7.3: Storage Subsystem Architecture Option 2

7.1.2 Plug-in Configuration

Plug-in developers **MUST** provide a plug-in configuration file in the following format:

```
[Groupname]
type=
name=
capability=
alive=
language=

queryrule=
sorttype=
```

Note: queryrule and sorttype are used only for framework storage plug-in configuration. The rest of the fields are for all plug-ins.

- Groupname is the group name for this configuration information. It collects the information below and assigns it to one plug-in. The Groupname can be regarded as a separator between two plug-ins' configuration information.
- type indicates the plug-in's type. It is an integer. (1 for transport plug-in, 2 for filter plug-in, 3 for framework storage plug-in, 4 for message storage plug-in)
- name is a string that represents the full path of the plug-in. It tells framework where to find the plug-in.

- **capability** indicates the capability of the plug-in. For transport and message storage plug-ins it indicates what message type this plug-in supports. Message types must be defined in the following manner: `mainType:subType`, with message types separated by comma. For filter plug-ins, it indicates what filter types the plug-in supports. Normally, it is a string list separated by commas.
- **alive** indicates whether or not the framework shall unload the plug-in after use. It is an integer. (0 for unload after use; 1 for always alive)
- **language** is a string value, indicating the programming language in which the plug-in is implemented. The supported values are `c` and `cpp`.
- **queryrule** is a string that indicates which query ids will be supported. Normally, it is a string list separated by commas.
- **sorttype** is a string that indicates which sort types will be supported. Normally, it is a string list separated by commas.

The following example defines the SMS message type `MSG_MESSAGE_TYPE_SMS` as 2, sub type `MSG_MESSAGE_TYPE_TEXT` is 0, `MSG_MESSAGE_TYPE_PORT` is 1.

```
[limo-transport-sms]
type=1
name=/usr/lib/libsmstransprovider.so
capability=2:0,2:1
alive=0
language=c
```

Suppose a filter plug-in supports the following filter ids: 1, 2, 3, 4, and 5. The configuration item should look like the following:

```
[limo-filter]
type=1
name=/usr/lib/libfilter.so
capability=1,2,3,4,5
alive=0
language=c
```

There is only one plug-in configuration file. All plug-ins' information shall be written there.

When the framework is initialized, it loads and parses this configuration file, then loads and manages all the required plug-ins according to the information in the file. The framework gets the capability of each plug-in, and dispatches messages, requests and other actions to the proper plug-in according to their types and capabilities.

7.1.2.1 Plug-in Interfaces

The following table describes the interfaces that plug-in developers must implement.

	Framework Storage (framework_plugin)	Message Plug-in (msg_plugin)
Message Filter		Msg_Plugin_Filter_Interface.h
Message Transport		Msg_Plugin_Transport_Interface.h
Storage, Message and Account	Msg_Framework_Storage_Interface.h	Msg_Plugin_Storage_Interface.h

7.2 Sequence Charts for the LiMo Messaging API

The following sequence charts are provided:

- [Filter Plug-in: Set Filter List to Filter Messages](#)

Legend

The chart below shows the legends used in the sequence charts.

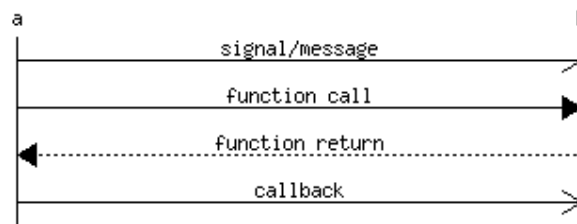


Figure 7.4: Sequence Chart Legend

Terms

- MAPI: Messaging API
- MsgFw: Messaging Framework
- TransportHandlerPlg: Transport Handler Plug-in
- FwStoPlg: Framework Storage Plug-in
- MsgStoPlg: Message Storage Plug-in
- FilterPlg: Filter plug-in

7.3 Filter Plug-in: Set Filter List to Filter Messages

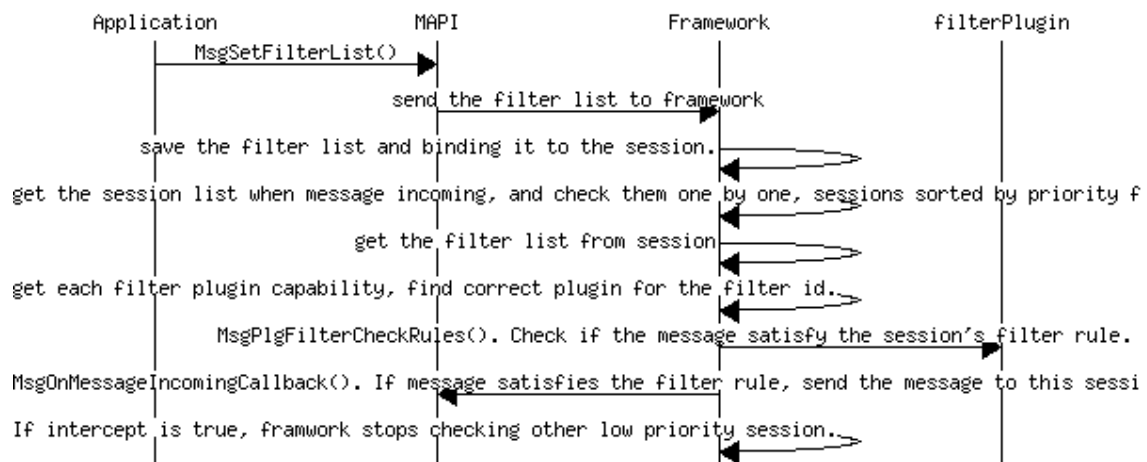


Figure 7.5: Filter Plug-in: Set Filter List to Filter Messages

Index

- [_MSG_ACCOUNT_ID_LIST_S, 9](#)
 - [accountIds, 9](#)
 - [iCount, 9](#)
- [_MSG_ACCOUNT_S, 10](#)
 - [accountId, 10](#)
 - [accountName, 10](#)
 - [defaultMsgStorageId, 10](#)
 - [extDataSize, 11](#)
 - [msgType, 10](#)
 - [pExtData, 11](#)
 - [rootFolderId, 10](#)
 - [storageId, 10](#)
- [_MSG_CONTACT_LIST_S, 12](#)
 - [iCount, 12](#)
 - [pContacts, 12](#)
- [_MSG_CONTACT_S, 13](#)
 - [contactName, 13](#)
 - [contactType, 13](#)
 - [contactVal, 13](#)
 - [extDataSize, 13](#)
 - [pExtData, 13](#)
- [_MSG_CONTACT_TYPE_E](#)
 - [Msg_Mapi_Types.h, 98](#)
- [_MSG_ENCODE_TYPE_E](#)
 - [Msg_Mapi_Types.h, 99](#)
- [_MSG_ERROR_E](#)
 - [Msg_Mapi_Types.h, 99](#)
- [_MSG_FILTER_ID_E](#)
 - [Msg_Mapi_FilterRule.h, 82](#)
- [_MSG_FILTER_ID_LIST_S, 15](#)
 - [filterIds, 15](#)
 - [iCount, 15](#)
- [_MSG_FILTER_LIST_S, 16](#)
 - [filters, 16](#)
 - [iCount, 16](#)
- [_MSG_FILTER_S, 17](#)
 - [autoStore, 17](#)
 - [filterId, 17](#)
 - [filterName, 17](#)
 - [filterValue, 18](#)
 - [intercept, 17](#)
 - [valueLength, 18](#)
- [_MSG_FOLDER_ID_LIST_S, 19](#)
 - [folderIds, 19](#)
 - [iCount, 19](#)
- [_MSG_FOLDER_S, 20](#)
 - [commExtDataSize, 20](#)
 - [folderId, 20](#)
 - [folderName, 20](#)
 - [folderType, 20](#)
 - [parentId, 20](#)
 - [pCommExtData, 21](#)
 - [sortType, 20](#)
 - [storageId, 20](#)
- [_MSG_FOLDER_TYPE_E](#)
 - [Msg_Mapi_Types.h, 100](#)
- [_MSG_FRAMEWORK_ERROR_E](#)
 - [Msg_Plugin_Types.h, 128](#)
- [_MSG_FW_STORAGE_S, 22](#)
 - [pAddAccount, 24](#)
 - [pAddFolder, 24](#)
 - [pAddMsg, 24](#)
 - [pAttachStorage, 24](#)
 - [pCheckStatus, 24](#)
 - [pCountAccount, 24](#)
 - [pCountMsg, 24](#)
 - [pCountSubFolder, 24](#)
 - [pCountUnreadMsg, 24](#)
 - [pDeleteAccount, 24](#)
 - [pDeleteAllMsg, 24](#)
 - [pDeleteFolder, 24](#)
 - [pDeleteMsg, 24](#)
 - [pDetachStorage, 24](#)
 - [pGetAccountIdsByRootId, 24](#)
 - [pGetAccountIdsByStorage, 24](#)
 - [pGetComAccount, 24](#)
 - [pGetDefaultAccID, 24](#)
 - [pGetFolder, 24](#)
 - [pGetFolderByType, 24](#)
 - [pGetFolderIdsByStorage, 24](#)
 - [pGetFolderParentId, 24](#)
 - [pGetMessageIdsByStorage, 24](#)
 - [pGetMsgCommonInfo, 24](#)
 - [pGetMsgParentId, 24](#)
 - [pGetStorageStatus, 24](#)
 - [pInitStorage, 24](#)
 - [pQueryAccount, 24](#)
 - [pQueryFolder, 24](#)
 - [pQueryMsg, 24](#)
 - [pReleaseAccountIdList, 24](#)

- pReleaseComAccount, 24
- pReleaseFolder, 24
- pReleaseFolderIdList, 24
- pReleaseMessageIdList, 24
- pReleaseMsgCommon, 24
- pSetDefaultAccID, 24
- pUpdateAccount, 24
- pUpdateFolder, 24
- pUpdateMsg, 24
- _MSG_MAIN_TYPE_E
 - Msg_Mapi_Types.h, 100
- _MSG_MEDIA_TYPE_E
 - Msg_Mapi_Types.h, 101
- _MSG_MESSAGE_ID_LIST_S, 26
 - iCount, 26
 - msgIds, 26
- _MSG_MESSAGE_PRIORITY_E
 - Msg_Mapi_Types.h, 101
- _MSG_MESSAGE_REQUEST_S, 27
 - extDataSize, 27
 - msgType, 27
 - pExtData, 28
 - pMsg, 27
 - reqId, 27
 - reqType, 27
- _MSG_MESSAGE_S, 29
 - accountId, 30
 - commExtDataSize, 31
 - deliveryStatus, 31
 - displayFrom, 30
 - displaySubject, 30
 - displayTime, 30
 - displayTo, 30
 - extDataSize, 31
 - from, 30
 - hasAttachment, 31
 - hasDrmContent, 31
 - msgId, 29
 - msgType, 29
 - parentId, 29
 - pCommExtData, 31
 - pExtData, 31
 - priority, 31
 - size, 31
 - status, 31
 - storageId, 30
 - subject, 30
 - to, 30
- _MSG_MESSAGE_STATUS_S, 33
 - error, 33
 - extDataSize, 33
 - isFinal, 33
 - msgId, 33
 - pExtData, 34
 - reqId, 33
 - status, 33
- _MSG_MESSAGE_TYPE_LIST_S, 35
 - iCount, 35
 - pMsgTypes, 35
- _MSG_MESSAGE_TYPE_S, 36
 - mainType, 36
 - subType, 36
- _MSG_PLG_ERROR_E
 - Msg_Plugin_Types.h, 128
- _MSG_PLG_STORAGE_S, 37
 - pAddAccount, 38
 - pAddMsg, 38
 - pAttachStorage, 38
 - pCheckStatus, 38
 - pDelAllMsg, 38
 - pDeleteAccount, 38
 - pDeleteMsg, 38
 - pDetachStorage, 38
 - pGetAccount, 38
 - pGetMsg, 38
 - pGetStorageStatus, 38
 - pInitStorage, 38
 - pReleaseAccount, 38
 - pReleaseMsg, 38
 - pSetFwStoHandle, 38
 - pUpdateAccount, 38
 - pUpdateMsg, 38
- _MSG_PLG_STORAGE_STATUS_E
 - Msg_Plugin_Types.h, 129
- _MSG_PLUGIN_FILTER_MANAGER_S, 39
 - pfCheckRules, 39
 - pfFinalize, 39
 - pfInitialize, 39
- _MSG_PLUGIN_TRANSPORT_S, 40
 - pfCancelRequest, 40
 - pfDiscardMsg, 40
 - pfFinalize, 40
 - pfInitialize, 40
 - pfRegisterListener, 40
 - pfSubmitRequest, 40
- _MSG_QUERY_ID_E
 - Msg_Mapi_QueryRule.h, 90
- _MSG_QUERY_ID_LIST_S, 41
 - iCount, 41
 - queryIds, 41
- _MSG_QUERY_LIST_S, 42
 - iCount, 42
 - queries, 42
- _MSG_QUERY_S, 43
 - queryId, 43
 - queryValue, 43
 - valueLength, 43
- _MSG_REQUEST_TYPE_E

- Msg_Mapi_Types.h, 101
- _MSG_SESSION_PRIORITY_E
 - Msg_Mapi_Types.h, 102
- _MSG_SIZE_UNIT_E
 - Msg_Mapi_Types.h, 102
- _MSG_SORT_RULE_S, 44
 - isAscending, 44
 - sortType, 44
- _MSG_SORT_TYPE_E
 - Msg_Mapi_QueryRule.h, 90
- _MSG_SORT_TYPE_LIST_S, 45
 - iCount, 45
 - sortTypes, 45
- _MSG_STATUS_E
 - Msg_Mapi_Types.h, 102
- _MSG_STORAGE_ID_LIST_S, 46
 - iCount, 46
 - pStorageIds, 46
- _MSG_STORAGE_ID_VALUE_E
 - Msg_Mapi_Types.h, 103
- _MSG_STRING_S, 47
 - encodeType, 47
 - length, 47
 - pString, 47
- _MSG_SUB_TYPE_E
 - Msg_Plugin_Types.h, 129
- _MSG_TRANSPORT_LISTENER_S, 48
 - pfReceiveMsgCallback, 48
 - pfStatusChangedCallback, 48
- accountId
 - _MSG_ACCOUNT_S, 10
 - _MSG_MESSAGE_S, 30
- accountIds
 - _MSG_ACCOUNT_ID_LIST_S, 9
- accountName
 - _MSG_ACCOUNT_S, 10
- autoStore
 - _MSG_FILTER_S, 17
- bool
 - Msg_Mapi_Types.h, 95
- commExtDataSize
 - _MSG_FOLDER_S, 20
 - _MSG_MESSAGE_S, 31
- COMPONENT_OVERVIEW_H
 - ComponentOverview.h, 49
- ComponentOverview.h, 49
 - COMPONENT_OVERVIEW_H, 49
- contactName
 - _MSG_CONTACT_S, 13
- contactType
 - _MSG_CONTACT_S, 13
- contactVal
 - _MSG_CONTACT_S, 13
- defaultMsgStorageId
 - _MSG_ACCOUNT_S, 10
- deliveryStatus
 - _MSG_MESSAGE_S, 31
- displayFrom
 - _MSG_MESSAGE_S, 30
- displaySubject
 - _MSG_MESSAGE_S, 30
- displayTime
 - _MSG_MESSAGE_S, 30
- displayTo
 - _MSG_MESSAGE_S, 30
- encodeType
 - _MSG_STRING_S, 47
- error
 - _MSG_MESSAGE_STATUS_S, 33
- extDataSize
 - _MSG_ACCOUNT_S, 11
 - _MSG_CONTACT_S, 13
 - _MSG_MESSAGE_REQUEST_S, 27
 - _MSG_MESSAGE_S, 31
 - _MSG_MESSAGE_STATUS_S, 33
- false
 - Msg_Mapi_Types.h, 95
- filterId
 - _MSG_FILTER_S, 17
- filterIds
 - _MSG_FILTER_ID_LIST_S, 15
- filterName
 - _MSG_FILTER_S, 17
- filters
 - _MSG_FILTER_LIST_S, 16
- filterValue
 - _MSG_FILTER_S, 18
- folderId
 - _MSG_FOLDER_S, 20
- folderIds
 - _MSG_FOLDER_ID_LIST_S, 19
- folderName
 - _MSG_FOLDER_S, 20
- folderType
 - _MSG_FOLDER_S, 20
- from
 - _MSG_MESSAGE_S, 30
- hasAttachment
 - _MSG_MESSAGE_S, 31
- hasDrmContent
 - _MSG_MESSAGE_S, 31

- iCount
 - [_MSG_ACCOUNT_ID_LIST_S](#), [9](#)
 - [_MSG_CONTACT_LIST_S](#), [12](#)
 - [_MSG_FILTER_ID_LIST_S](#), [15](#)
 - [_MSG_FILTER_LIST_S](#), [16](#)
 - [_MSG_FOLDER_ID_LIST_S](#), [19](#)
 - [_MSG_MESSAGE_ID_LIST_S](#), [26](#)
 - [_MSG_MESSAGE_TYPE_LIST_S](#), [35](#)
 - [_MSG_QUERY_ID_LIST_S](#), [41](#)
 - [_MSG_QUERY_LIST_S](#), [42](#)
 - [_MSG_SORT_TYPE_LIST_S](#), [45](#)
 - [_MSG_STORAGE_ID_LIST_S](#), [46](#)
- intercept
 - [_MSG_FILTER_S](#), [17](#)
- isAscending
 - [_MSG_SORT_RULE_S](#), [44](#)
- isFinal
 - [_MSG_MESSAGE_STATUS_S](#), [33](#)
- length
 - [_MSG_STRING_S](#), [47](#)
- mainType
 - [_MSG_MESSAGE_TYPE_S](#), [36](#)
- MSG_CONTACT_TYPE_EMAIL
 - [Msg_Mapi_Types.h](#), [98](#)
- MSG_CONTACT_TYPE_PLMN
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_CONTACT_TYPE_UNKNOWN
 - [Msg_Mapi_Types.h](#), [98](#)
- MSG_ENCODE_ASCII
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ENCODE_GSM7BIT
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ENCODE_SHIFT_JIS
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ENCODE_UCS2
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ENCODE_UTF16
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ENCODE_UTF8
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_COMMUNICATION_ERROR
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_CONNECTION_NOT_CONNECT
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_FRAMEWORK_RESERVE_BEGIN
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_FRAMEWORK_RESERVE_END
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_FS_ERROR
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_FS_FULL
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_ACCOUNT
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_INVALID_ACCOUNT_ID
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_CONNECTION
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_FILTER_ID
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_FOLDER
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_INVALID_FOLDER_ID
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_FOLDER_TYPE
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_INVALID_MESSAGE
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_INVALID_MESSAGE_ID
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_MESSAGE_UID
 - [Msg_Plugin_Types.h](#), [129](#)
- MSG_ERR_INVALID_MSG_TYPE
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_PLUGIN_HANDLE
 - [Msg_Plugin_Types.h](#), [128](#)
- MSG_ERR_INVALID_QUERY_ID
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_REQUEST
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_REQUEST_ID
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_INVALID_SESSION_HANDLE
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_SESSION_PRIORITY
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_SORT_TYPE
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_STORAGE
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_STORAGE_REG
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_INVALID_TRANS_HANDLE
 - [Msg_Plugin_Types.h](#), [129](#)
- MSG_ERR_MAX_NUMBER_REACHED
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_MEMORY_ERROR
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_NOT_EMPTY_FOLDER
 - [Msg_Mapi_Types.h](#), [100](#)
- MSG_ERR_NULL_POINTER
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_PERMISSION_DENIED
 - [Msg_Mapi_Types.h](#), [99](#)
- MSG_ERR_PLUGIN
 - [Msg_Mapi_Types.h](#), [100](#)

- MSG_ERR_PLUGIN_BUSY
 - Msg_Plugin_Types.h, [129](#)
- MSG_ERR_PLUGIN_DEF_BEGIN_VALUE
 - Msg_Mapi_Types.h, [100](#)
- MSG_ERR_ROOTFOLDER_INUSE
 - Msg_Mapi_Types.h, [100](#)
- MSG_ERR_SESSION_NOT_CONNECTED
 - Msg_Mapi_Types.h, [99](#)
- MSG_ERR_STORAGE_ERROR
 - Msg_Mapi_Types.h, [100](#)
- MSG_ERR_STORAGE_FULL
 - Msg_Mapi_Types.h, [100](#)
- MSG_ERR_STORAGE_NOT_INIT
 - Msg_Mapi_Types.h, [99](#)
- MSG_ERR_STORAGE_REG_NOT_INIT
 - Msg_Mapi_Types.h, [99](#)
- MSG_ERR_UNKNOWN
 - Msg_Mapi_Types.h, [100](#)
- MSG_FILTER_BY_FROM
 - Msg_Mapi_FilterRule.h, [82](#)
- MSG_FILTER_BY_HAS_ATTACHMENT
 - Msg_Mapi_FilterRule.h, [82](#)
- MSG_FILTER_BY_HAS_DRM_CONTENT
 - Msg_Mapi_FilterRule.h, [82](#)
- MSG_FILTER_BY_MSG_TYPE
 - Msg_Mapi_FilterRule.h, [82](#)
- MSG_FILTER_BY_PRIORITY_GT
 - Msg_Mapi_FilterRule.h, [83](#)
- MSG_FILTER_BY_PRIORITY_LT
 - Msg_Mapi_FilterRule.h, [83](#)
- MSG_FILTER_BY_SIZE_GT
 - Msg_Mapi_FilterRule.h, [83](#)
- MSG_FILTER_BY_SIZE_LT
 - Msg_Mapi_FilterRule.h, [83](#)
- MSG_FILTER_BY_UNKNOWN
 - Msg_Mapi_FilterRule.h, [82](#)
- MSG_FILTER_BY_USER_DEF_BEGIN
 - Msg_Mapi_FilterRule.h, [83](#)
- MSG_FOLDER_TYPE_DELETED
 - Msg_Mapi_Types.h, [100](#)
- MSG_FOLDER_TYPE_DRAFTS
 - Msg_Mapi_Types.h, [100](#)
- MSG_FOLDER_TYPE_INBOX
 - Msg_Mapi_Types.h, [100](#)
- MSG_FOLDER_TYPE_OUTBOX
 - Msg_Mapi_Types.h, [100](#)
- MSG_FOLDER_TYPE_ROOT
 - Msg_Mapi_Types.h, [100](#)
- MSG_FOLDER_TYPE_SENT
 - Msg_Mapi_Types.h, [100](#)
- MSG_FOLDER_TYPE_USERDEFINE
 - Msg_Mapi_Types.h, [100](#)
- Msg_Mapi_FilterRule.h
 - MSG_FILTER_BY_FROM, [82](#)
 - MSG_FILTER_BY_HAS_ATTACHMENT, [82](#)
 - MSG_FILTER_BY_HAS_DRM_CONTENT, [82](#)
 - MSG_FILTER_BY_MSG_TYPE, [82](#)
 - MSG_FILTER_BY_PRIORITY_GT, [83](#)
 - MSG_FILTER_BY_PRIORITY_LT, [83](#)
 - MSG_FILTER_BY_SIZE_GT, [83](#)
 - MSG_FILTER_BY_SIZE_LT, [83](#)
 - MSG_FILTER_BY_UNKNOWN, [82](#)
 - MSG_FILTER_BY_USER_DEF_BEGIN, [83](#)
- Msg_Mapi_QueryRule.h
 - MSG_QUERY_BY_AFTER_DATETIME, [90](#)
 - MSG_QUERY_BY_BEFORE_DATETIME, [90](#)
 - MSG_QUERY_BY_DELIVERY_STATUS, [90](#)
 - MSG_QUERY_BY_FROM, [90](#)
 - MSG_QUERY_BY_HASATTACHMENT, [90](#)
 - MSG_QUERY_BY_HASDRMCONTENT, [90](#)
 - MSG_QUERY_BY_PRIORITY, [90](#)
 - MSG_QUERY_BY_SIZE, [90](#)
 - MSG_QUERY_BY_STATUS, [90](#)
 - MSG_QUERY_BY_SUBJECT, [90](#)
 - MSG_QUERY_BY_TO, [90](#)
 - MSG_QUERY_BY_UNKNOWN, [90](#)
 - MSG_QUERY_BY_USER_DEF_BEGIN, [90](#)
 - MSG_SORT_BY_DELIVERY_STATUS, [90](#)
 - MSG_SORT_BY_DISPLAY_FROM, [90](#)
 - MSG_SORT_BY_DISPLAY_SUBJECT, [90](#)
 - MSG_SORT_BY_DISPLAY_TIME, [90](#)
 - MSG_SORT_BY_DISPLAY_TO, [90](#)
 - MSG_SORT_BY_HASATTACHMENT, [90](#)
 - MSG_SORT_BY_HASDRMCONTENT, [90](#)
 - MSG_SORT_BY_PRIORITY, [90](#)
 - MSG_SORT_BY_SIZE, [90](#)
 - MSG_SORT_BY_STATUS, [90](#)
 - MSG_SORT_BY_UNKNOWN, [90](#)
 - MSG_SORT_BY_USER_DEF_BEGIN, [90](#)
- Msg_Mapi_Types.h
 - MSG_CONTACT_TYPE_EMAIL, [98](#)
 - MSG_CONTACT_TYPE_PLMN, [99](#)
 - MSG_CONTACT_TYPE_UNKNOWN, [98](#)
 - MSG_ENCODE_ASCII, [99](#)
 - MSG_ENCODE_GSM7BIT, [99](#)
 - MSG_ENCODE_SHIFT_JIS, [99](#)
 - MSG_ENCODE_UCS2, [99](#)
 - MSG_ENCODE_UTF16, [99](#)
 - MSG_ENCODE_UTF8, [99](#)
 - MSG_ERR_COMMUNICATION_ERROR, [100](#)
 - MSG_ERR_CONNECTION_NOT_CONNECT, [100](#)

- MSG_ERR_FRAMEWORK_RESERVE_-
BEGIN, [100](#)
- MSG_ERR_FRAMEWORK_RESERVE_-
END, [100](#)
- MSG_ERR_FS_ERROR, [99](#)
- MSG_ERR_FS_FULL, [99](#)
- MSG_ERR_INVALID_ACCOUNT, [100](#)
- MSG_ERR_INVALID_ACCOUNT_ID, [99](#)
- MSG_ERR_INVALID_CONNECTION, [99](#)
- MSG_ERR_INVALID_FILTER_ID, [99](#)
- MSG_ERR_INVALID_FOLDER, [100](#)
- MSG_ERR_INVALID_FOLDER_ID, [99](#)
- MSG_ERR_INVALID_FOLDER_TYPE, [100](#)
- MSG_ERR_INVALID_MESSAGE, [100](#)
- MSG_ERR_INVALID_MESSAGE_ID, [99](#)
- MSG_ERR_INVALID_MSG_TYPE, [99](#)
- MSG_ERR_INVALID_QUERY_ID, [99](#)
- MSG_ERR_INVALID_REQUEST, [99](#)
- MSG_ERR_INVALID_REQUEST_ID, [100](#)
- MSG_ERR_INVALID_SESSION_HANDLE,
[99](#)
- MSG_ERR_INVALID_SESSION_-
PRIORITY, [99](#)
- MSG_ERR_INVALID_SORT_TYPE, [99](#)
- MSG_ERR_INVALID_STORAGE, [99](#)
- MSG_ERR_INVALID_STORAGE_REG, [99](#)
- MSG_ERR_MAX_NUMBER_REACHED,
[100](#)
- MSG_ERR_MEMORY_ERROR, [100](#)
- MSG_ERR_NOT_EMPTY_FOLDER, [100](#)
- MSG_ERR_NULL_POINTER, [99](#)
- MSG_ERR_PERMISSION_DENIED, [99](#)
- MSG_ERR_PLUGIN, [100](#)
- MSG_ERR_PLUGIN_DEF_BEGIN_VALUE,
[100](#)
- MSG_ERR_ROOTFOLDER_INUSE, [100](#)
- MSG_ERR_SESSION_NOT_CONNECTED,
[99](#)
- MSG_ERR_STORAGE_ERROR, [100](#)
- MSG_ERR_STORAGE_FULL, [100](#)
- MSG_ERR_STORAGE_NOT_INIT, [99](#)
- MSG_ERR_STORAGE_REG_NOT_INIT, [99](#)
- MSG_ERR_UNKNOWN, [100](#)
- MSG_FOLDER_TYPE_DELETED, [100](#)
- MSG_FOLDER_TYPE_DRAFTS, [100](#)
- MSG_FOLDER_TYPE_INBOX, [100](#)
- MSG_FOLDER_TYPE_OUTBOX, [100](#)
- MSG_FOLDER_TYPE_ROOT, [100](#)
- MSG_FOLDER_TYPE_SENT, [100](#)
- MSG_FOLDER_TYPE_USERDEFINE, [100](#)
- MSG_MEDIA_FLASH, [101](#)
- MSG_MEDIA_REMOVABLE, [101](#)
- MSG_MEDIA_SIM, [101](#)
- MSG_MESSAGE_PRIORITY_HIGH, [101](#)
- MSG_MESSAGE_PRIORITY_LOW, [101](#)
- MSG_MESSAGE_PRIORITY_NORMAL,
[101](#)
- MSG_MESSAGE_TYPE_EMAIL, [101](#)
- MSG_MESSAGE_TYPE_IMODE, [101](#)
- MSG_MESSAGE_TYPE_MMS, [101](#)
- MSG_MESSAGE_TYPE_SMS, [101](#)
- MSG_MESSAGE_TYPE_UNKNOWN, [101](#)
- MSG_REQUEST_TYPE_INVALID, [101](#)
- MSG_REQUEST_TYPE_RESERVED_-
BEGIN, [101](#)
- MSG_REQUEST_TYPE_RESERVED_END,
[102](#)
- MSG_REQUEST_TYPE_SEND, [101](#)
- MSG_REQUEST_TYPE_SENDSAVE, [101](#)
- MSG_REQUEST_TYPE_USER_DEF_-
BEGIN, [102](#)
- MSG_SESSION_PRIORITY_HIGH, [102](#)
- MSG_SESSION_PRIORITY_HIGHEST, [102](#)
- MSG_SESSION_PRIORITY_LOW, [102](#)
- MSG_SESSION_PRIORITY_LOWEST, [102](#)
- MSG_SESSION_PRIORITY_NORMAL, [102](#)
- MSG_SIZE_UNIT_BYTE, [102](#)
- MSG_SIZE_UNIT_GBYTE, [102](#)
- MSG_SIZE_UNIT_KBYTE, [102](#)
- MSG_SIZE_UNIT_MBYTE, [102](#)
- MSG_SIZE_UNIT_NUMBER, [102](#)
- MSG_STATUS_DRAFT, [103](#)
- MSG_STATUS_EDITING, [103](#)
- MSG_STATUS_FAIL_TO_SEND, [103](#)
- MSG_STATUS_FAIL_TO_SENDSAVE, [103](#)
- MSG_STATUS_FORWARDED, [103](#)
- MSG_STATUS_PROTECTED, [103](#)
- MSG_STATUS_READ, [102](#)
- MSG_STATUS_REPLIED, [103](#)
- MSG_STATUS_REQ_CANCELLED, [103](#)
- MSG_STATUS_SAVED_NOTSENT, [103](#)
- MSG_STATUS_SENDING, [103](#)
- MSG_STATUS_SENT, [103](#)
- MSG_STATUS_SENT_NOTSAVED, [103](#)
- MSG_STATUS_SENT_SAVED, [103](#)
- MSG_STATUS_UNKNOWN, [102](#)
- MSG_STATUS_UNREAD, [103](#)
- MSG_STATUS_USER_DEF_BEGIN, [103](#)
- MSG_STORAGE_DEFAULT, [103](#)
- MSG_STORAGE_UNKNOWN, [103](#)
- MSG_SUCCESS, [99](#)
- MSG_MEDIA_FLASH
Msg_Mapi_Types.h, [101](#)
- MSG_MEDIA_REMOVABLE
Msg_Mapi_Types.h, [101](#)
- MSG_MEDIA_SIM
Msg_Mapi_Types.h, [101](#)
- MSG_MESSAGE_PRIORITY_HIGH

- Msg_Mapi_Types.h, [101](#)
- MSG_MESSAGE_PRIORITY_LOW
 - Msg_Mapi_Types.h, [101](#)
- MSG_MESSAGE_PRIORITY_NORMAL
 - Msg_Mapi_Types.h, [101](#)
- MSG_MESSAGE_TYPE_EMAIL
 - Msg_Mapi_Types.h, [101](#)
- MSG_MESSAGE_TYPE_IMODE
 - Msg_Mapi_Types.h, [101](#)
- MSG_MESSAGE_TYPE_MMS
 - Msg_Mapi_Types.h, [101](#)
- MSG_MESSAGE_TYPE_SMS
 - Msg_Mapi_Types.h, [101](#)
- MSG_MESSAGE_TYPE_UNKNOWN
 - Msg_Mapi_Types.h, [101](#)
- MSG_MSG_TYPE_SMS_TEXT
 - Msg_Plugin_Types.h, [129](#)
- MSG_PLG_DATA_AVAILABLE
 - Msg_Plugin_Types.h, [129](#)
- MSG_PLG_DATA_NOT_AVAILABLE
 - Msg_Plugin_Types.h, [129](#)
- MSG_PLG_MEDIA_NOT_AVAILABLE
 - Msg_Plugin_Types.h, [129](#)
- Msg_Plugin_Types.h
 - MSG_ERR_INVALID_MESSAGE_UID, [129](#)
 - MSG_ERR_INVALID_PLUGIN_HANDLE, [128](#)
 - MSG_ERR_INVALID_TRANS_HANDLE, [129](#)
 - MSG_ERR_PLUGIN_BUSY, [129](#)
 - MSG_MSG_TYPE_SMS_TEXT, [129](#)
 - MSG_PLG_DATA_AVAILABLE, [129](#)
 - MSG_PLG_DATA_NOT_AVAILABLE, [129](#)
 - MSG_PLG_MEDIA_NOT_AVAILABLE, [129](#)
- MSG_QUERY_BY_AFTER_DATETIME
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_BEFORE_DATETIME
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_DELIVERY_STATUS
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_FROM
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_HASATTACHMENT
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_HASDRMCONTENT
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_PRIORITY
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_SIZE
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_STATUS
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_SUBJECT
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_TO
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_UNKNOWN
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_QUERY_BY_USER_DEF_BEGIN
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_REQUEST_TYPE_INVALID
 - Msg_Mapi_Types.h, [101](#)
- MSG_REQUEST_TYPE_RESERVED_BEGIN
 - Msg_Mapi_Types.h, [101](#)
- MSG_REQUEST_TYPE_RESERVED_END
 - Msg_Mapi_Types.h, [102](#)
- MSG_REQUEST_TYPE_SEND
 - Msg_Mapi_Types.h, [101](#)
- MSG_REQUEST_TYPE_SENDSAVE
 - Msg_Mapi_Types.h, [101](#)
- MSG_REQUEST_TYPE_USER_DEF_BEGIN
 - Msg_Mapi_Types.h, [102](#)
- MSG_SESSION_PRIORITY_HIGH
 - Msg_Mapi_Types.h, [102](#)
- MSG_SESSION_PRIORITY_HIGHEST
 - Msg_Mapi_Types.h, [102](#)
- MSG_SESSION_PRIORITY_LOW
 - Msg_Mapi_Types.h, [102](#)
- MSG_SESSION_PRIORITY_LOWEST
 - Msg_Mapi_Types.h, [102](#)
- MSG_SESSION_PRIORITY_NORMAL
 - Msg_Mapi_Types.h, [102](#)
- MSG_SIZE_UNIT_BYTE
 - Msg_Mapi_Types.h, [102](#)
- MSG_SIZE_UNIT_GBYTE
 - Msg_Mapi_Types.h, [102](#)
- MSG_SIZE_UNIT_KBYTE
 - Msg_Mapi_Types.h, [102](#)
- MSG_SIZE_UNIT_MBYTE
 - Msg_Mapi_Types.h, [102](#)
- MSG_SIZE_UNIT_NUMBER
 - Msg_Mapi_Types.h, [102](#)
- MSG_SORT_BY_DELIVERY_STATUS
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_SORT_BY_DISPLAY_FROM
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_SORT_BY_DISPLAY_SUBJECT
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_SORT_BY_DISPLAY_TIME
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_SORT_BY_DISPLAY_TO
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_SORT_BY_HASATTACHMENT
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_SORT_BY_HASDRMCONTENT
 - Msg_Mapi_QueryRule.h, [90](#)
- MSG_SORT_BY_PRIORITY

- Msg_Mapi_QueryRule.h, 90
- MSG_SORT_BY_SIZE
 - Msg_Mapi_QueryRule.h, 90
- MSG_SORT_BY_STATUS
 - Msg_Mapi_QueryRule.h, 90
- MSG_SORT_BY_UNKNOWN
 - Msg_Mapi_QueryRule.h, 90
- MSG_SORT_BY_USER_DEF_BEGIN
 - Msg_Mapi_QueryRule.h, 90
- MSG_STATUS_DRAFT
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_EDITING
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_FAIL_TO_SEND
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_FAIL_TO_SENDSAVE
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_FORWARDED
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_PROTECTED
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_READ
 - Msg_Mapi_Types.h, 102
- MSG_STATUS_REPLIED
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_REQ_CANCELLED
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_SAVED_NOTSENT
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_SENDING
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_SENT
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_SENT_NOTSAVED
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_SENT_SAVED
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_UNKNOWN
 - Msg_Mapi_Types.h, 102
- MSG_STATUS_UNREAD
 - Msg_Mapi_Types.h, 103
- MSG_STATUS_USER_DEF_BEGIN
 - Msg_Mapi_Types.h, 103
- MSG_STORAGE_DEFAULT
 - Msg_Mapi_Types.h, 103
- MSG_STORAGE_UNKNOWN
 - Msg_Mapi_Types.h, 103
- MSG_SUCCESS
 - Msg_Mapi_Types.h, 99
- MSG_ACCOUNT_ID_LIST_S
 - Msg_Mapi_Types.h, 96
- MSG_ACCOUNT_ID_T
 - Msg_Mapi_Types.h, 96
- MSG_ACCOUNT_S
 - Msg_Mapi_Account.h, 79
- MSG_CONTACT_LIST_S
 - Msg_Mapi_Contact.h, 80
- MSG_CONTACT_S
 - Msg_Mapi_Contact.h, 80
- MSG_CONTACT_TYPE_T
 - Msg_Mapi_Types.h, 96
- MSG_DEFAULT_ACCOUNT_ID
 - Msg_Mapi_Types.h, 95
- MSG_DEFAULT_FOLDER_ID
 - Msg_Mapi_Types.h, 95
- MSG_ENCODE_TYPE_T
 - Msg_Mapi_Types.h, 96
- MSG_ERROR_BEGIN
 - Msg_Mapi_Types.h, 95
- MSG_ERROR_T
 - Msg_Mapi_Types.h, 96
- MSG_FILTER_ID_LIST_S
 - Msg_Mapi_FilterRule.h, 82
- MSG_FILTER_ID_T
 - Msg_Mapi_FilterRule.h, 82
- MSG_FILTER_LIST_S
 - Msg_Mapi_FilterRule.h, 82
- MSG_FILTER_S
 - Msg_Mapi_FilterRule.h, 82
- MSG_FOLDER_ID_LIST_S
 - Msg_Mapi_Types.h, 96
- MSG_FOLDER_ID_T
 - Msg_Mapi_Types.h, 97
- MSG_FOLDER_S
 - Msg_Mapi_Folder.h, 84
- MSG_FOLDER_TYPE_T
 - Msg_Mapi_Types.h, 97
- Msg_Framework_Storage_Interface.h, 50
- MSG_FW_STORAGE_S, 53
 - MsgFwStoAddAccount, 53
 - MsgFwStoAddFolder, 54
 - MsgFwStoAddMessage, 55
 - MsgFwStoAttachStorage, 55
 - MsgFwStoCheckStatus, 56
 - MsgFwStoCountAccount, 56
 - MsgFwStoCountMessage, 57
 - MsgFwStoCountSubFolder, 58
 - MsgFwStoCountUnreadMessage, 58
 - MsgFwStoCreateHandle, 78
 - MsgFwStoDelAllMsgInFolder, 59
 - MsgFwStoDeleteAccount, 60
 - MsgFwStoDeleteFolder, 60
 - MsgFwStoDeleteMessage, 61
 - MsgFwStoDestroyHandle, 78
 - MsgFwStoDetachStorage, 62
 - MsgFwStoGetAccountIdsByRootId, 62
 - MsgFwStoGetAccountIdsByStorage, 63
 - MsgFwStoGetCommAccount, 64

- MsgFwStoGetDefaultAccID, 64
- MsgFwStoGetFolder, 65
- MsgFwStoGetFolderByType, 65
- MsgFwStoGetFolderIdsByStorage, 66
- MsgFwStoGetFolderParentId, 67
- MsgFwStoGetMessageIdsByStorage, 67
- MsgFwStoGetMessageParentId, 68
- MsgFwStoGetMsgCommInfo, 68
- MsgFwStoGetStorageStatus, 69
- MsgFwStoInitStorage, 70
- MsgFwStoQueryAccount, 70
- MsgFwStoQueryFolder, 71
- MsgFwStoQueryMessage, 72
- MsgFwStoReleaseAccountIdList, 73
- MsgFwStoReleaseCommAccount, 73
- MsgFwStoReleaseFolder, 73
- MsgFwStoReleaseFolderIdList, 74
- MsgFwStoReleaseMessageList, 74
- MsgFwStoReleaseMsgCommon, 74
- MsgFwStoSetDefaultAccID, 75
- MsgFwStoUpdateAccount, 75
- MsgFwStoUpdateFolder, 76
- MsgFwStoUpdateMessage, 77
- MSG_FW_STORAGE_S
 - Msg_Framework_Storage_Interface.h, 53
- MSG_INVALID_ACCOUNT_ID
 - Msg_Mapi_Types.h, 96
- MSG_INVALID_FOLDER_ID
 - Msg_Mapi_Types.h, 96
- MSG_INVALID_MESSAGE_ID
 - Msg_Mapi_Types.h, 96
- MSG_MAIN_TYPE_T
 - Msg_Mapi_Types.h, 97
- Msg_Mapi_Account.h, 79
 - MSG_ACCOUNT_S, 79
- Msg_Mapi_Contact.h, 80
 - MSG_CONTACT_LIST_S, 80
 - MSG_CONTACT_S, 80
- Msg_Mapi_FilterRule.h, 81
 - _MSG_FILTER_ID_E, 82
 - MSG_FILTER_ID_LIST_S, 82
 - MSG_FILTER_ID_T, 82
 - MSG_FILTER_LIST_S, 82
 - MSG_FILTER_S, 82
- Msg_Mapi_Folder.h, 84
 - MSG_FOLDER_S, 84
- Msg_Mapi_Message.h, 85
 - MSG_MESSAGE_S, 85
- Msg_Mapi_Message_Request.h, 86
 - MSG_MESSAGE_REQUEST_S, 86
- Msg_Mapi_Message_Status.h, 87
 - MSG_MESSAGE_STATUS_S, 87
- Msg_Mapi_QueryRule.h, 88
 - _MSG_QUERY_ID_E, 90
 - _MSG_SORT_TYPE_E, 90
 - MSG_QUERY_ID_LIST_S, 89
 - MSG_QUERY_ID_T, 89
 - MSG_QUERY_LIST_S, 89
 - MSG_QUERY_S, 89
 - MSG_SORT_RULE_S, 89
 - MSG_SORT_TYPE_LIST_S, 89
 - MSG_SORT_TYPE_T, 89
- Msg_Mapi_Types.h, 91
 - _MSG_CONTACT_TYPE_E, 98
 - _MSG_ENCODE_TYPE_E, 99
 - _MSG_ERROR_E, 99
 - _MSG_FOLDER_TYPE_E, 100
 - _MSG_MAIN_TYPE_E, 100
 - _MSG_MEDIA_TYPE_E, 101
 - _MSG_MESSAGE_PRIORITY_E, 101
 - _MSG_REQUEST_TYPE_E, 101
 - _MSG_SESSION_PRIORITY_E, 102
 - _MSG_SIZE_UNIT_E, 102
 - _MSG_STATUS_E, 102
 - _MSG_STORAGE_ID_VALUE_E, 103
 - bool, 95
 - false, 95
 - MSG_ACCOUNT_ID_LIST_S, 96
 - MSG_ACCOUNT_ID_T, 96
 - MSG_CONTACT_TYPE_T, 96
 - MSG_DEFAULT_ACCOUNT_ID, 95
 - MSG_DEFAULT_FOLDER_ID, 95
 - MSG_ENCODE_TYPE_T, 96
 - MSG_ERROR_BEGIN, 95
 - MSG_ERROR_T, 96
 - MSG_FOLDER_ID_LIST_S, 96
 - MSG_FOLDER_ID_T, 97
 - MSG_FOLDER_TYPE_T, 97
 - MSG_INVALID_ACCOUNT_ID, 96
 - MSG_INVALID_FOLDER_ID, 96
 - MSG_INVALID_MESSAGE_ID, 96
 - MSG_MAIN_TYPE_T, 97
 - MSG_MEDIA_TYPE_T, 97
 - MSG_MESSAGE_ID_LIST_S, 97
 - MSG_MESSAGE_ID_T, 97
 - MSG_MESSAGE_PRIORITY_T, 97
 - MSG_MESSAGE_TYPE_LIST_S, 97
 - MSG_MESSAGE_TYPE_S, 97
 - MSG_PLUGIN_ERROR_T, 97
 - MSG_REQUEST_ID_T, 97
 - MSG_REQUEST_TYPE_T, 97
 - MSG_SESSION_PRIORITY_T, 98
 - MSG_SIZE_UNIT_T, 98
 - MSG_STATUS_T, 98
 - MSG_STORAGE_ID_LIST_S, 98
 - MSG_STORAGE_ID_T, 98
 - MSG_STRING_S, 98
 - MSG_SUB_TYPE_T, 98

- true, 96
- MSG_MEDIA_TYPE_T
 - Msg_Mapi_Types.h, 97
- MSG_MESSAGE_ID_LIST_S
 - Msg_Mapi_Types.h, 97
- MSG_MESSAGE_ID_T
 - Msg_Mapi_Types.h, 97
- MSG_MESSAGE_PRIORITY_T
 - Msg_Mapi_Types.h, 97
- MSG_MESSAGE_REQUEST_S
 - Msg_Mapi_Message_Request.h, 86
- MSG_MESSAGE_S
 - Msg_Mapi_Message.h, 85
- MSG_MESSAGE_STATUS_S
 - Msg_Mapi_Message_Status.h, 87
- MSG_MESSAGE_TYPE_LIST_S
 - Msg_Mapi_Types.h, 97
- MSG_MESSAGE_TYPE_S
 - Msg_Mapi_Types.h, 97
- MSG_PLG_STORAGE_S
 - Msg_Plugin_Storage_Interface.h, 109
- MSG_PLG_STORAGE_STATUS_T
 - Msg_Plugin_Types.h, 128
- MSG_PLUGIN_ERROR_T
 - Msg_Mapi_Types.h, 97
- Msg_Plugin_Filter_Interface.h, 104
 - MSG_PLUGIN_FILTER_MANAGER_S, 104
 - MsgPlgFilterCheckRules, 104
 - MsgPlgFilterCreateHandle, 106
 - MsgPlgFilterDestroyHandle, 106
 - MsgPlgFilterFinalize, 105
 - MsgPlgFilterInitialize, 105
- MSG_PLUGIN_FILTER_MANAGER_S
 - Msg_Plugin_Filter_Interface.h, 104
- Msg_Plugin_Storage_Interface.h, 107
 - MSG_PLG_STORAGE_S, 109
 - MsgPlgStoAddAccount, 109
 - MsgPlgStoAddMessage, 109
 - MsgPlgStoAttachStorage, 110
 - MsgPlgStoCheckStatus, 111
 - MsgPlgStoCreateHandle, 119
 - MsgPlgStoDelAllMsgInFolder, 111
 - MsgPlgStoDeleteAccount, 112
 - MsgPlgStoDeleteMessage, 112
 - MsgPlgStoDestroyHandle, 120
 - MsgPlgStoDetachStorage, 113
 - MsgPlgStoGetAccount, 114
 - MsgPlgStoGetMessage, 114
 - MsgPlgStoGetStorageStatus, 115
 - MsgPlgStoInitStorage, 116
 - MsgPlgStoReleaseAccount, 116
 - MsgPlgStoReleaseMessage, 117
 - MsgPlgStoSetFwStoHandle, 117
 - MsgPlgStoUpdateAccount, 118
 - MsgPlgStoUpdateMessage, 118
- Msg_Plugin_Transport_Interface.h, 121
 - MSG_PLUGIN_TRANSPORT_S, 122
 - MSG_TRANSPORT_LISTENER_S, 122
 - MsgPlgTransCancelRequest, 122
 - MsgPlgTransCreateHandle, 126
 - MsgPlgTransDestroyHandle, 126
 - MsgPlgTransDiscardMsg, 122
 - MsgPlgTransFinalize, 123
 - MsgPlgTransInitialize, 123
 - MsgPlgTransOnMsgIncoming, 123
 - MsgPlgTransOnStatusChanged, 124
 - MsgPlgTransRegisterListener, 125
 - MsgPlgTransSubmitRequest, 125
- MSG_PLUGIN_TRANSPORT_S
 - Msg_Plugin_Transport_Interface.h, 122
- Msg_Plugin_Types.h, 128
 - _MSG_FRAMEWORK_ERROR_E, 128
 - _MSG_PLG_ERROR_E, 128
 - _MSG_PLG_STORAGE_STATUS_E, 129
 - _MSG_SUB_TYPE_E, 129
 - MSG_PLG_STORAGE_STATUS_T, 128
- MSG_QUERY_ID_LIST_S
 - Msg_Mapi_QueryRule.h, 89
- MSG_QUERY_ID_T
 - Msg_Mapi_QueryRule.h, 89
- MSG_QUERY_LIST_S
 - Msg_Mapi_QueryRule.h, 89
- MSG_QUERY_S
 - Msg_Mapi_QueryRule.h, 89
- MSG_REQUEST_ID_T
 - Msg_Mapi_Types.h, 97
- MSG_REQUEST_TYPE_T
 - Msg_Mapi_Types.h, 97
- MSG_SESSION_PRIORITY_T
 - Msg_Mapi_Types.h, 98
- MSG_SIZE_UNIT_T
 - Msg_Mapi_Types.h, 98
- MSG_SORT_RULE_S
 - Msg_Mapi_QueryRule.h, 89
- MSG_SORT_TYPE_LIST_S
 - Msg_Mapi_QueryRule.h, 89
- MSG_SORT_TYPE_T
 - Msg_Mapi_QueryRule.h, 89
- MSG_STATUS_T
 - Msg_Mapi_Types.h, 98
- MSG_STORAGE_ID_LIST_S
 - Msg_Mapi_Types.h, 98
- MSG_STORAGE_ID_T
 - Msg_Mapi_Types.h, 98
- MSG_STRING_S
 - Msg_Mapi_Types.h, 98
- MSG_SUB_TYPE_T
 - Msg_Mapi_Types.h, 98

- MSG_TRANSPORT_LISTENER_S
 - Msg_Plugin_Transport_Interface.h, 122
- MsgFwStoAddAccount
 - Msg_Framework_Storage_Interface.h, 53
- MsgFwStoAddFolder
 - Msg_Framework_Storage_Interface.h, 54
- MsgFwStoAddMessage
 - Msg_Framework_Storage_Interface.h, 55
- MsgFwStoAttachStorage
 - Msg_Framework_Storage_Interface.h, 55
- MsgFwStoCheckStatus
 - Msg_Framework_Storage_Interface.h, 56
- MsgFwStoCountAccount
 - Msg_Framework_Storage_Interface.h, 56
- MsgFwStoCountMessage
 - Msg_Framework_Storage_Interface.h, 57
- MsgFwStoCountSubFolder
 - Msg_Framework_Storage_Interface.h, 58
- MsgFwStoCountUnreadMessage
 - Msg_Framework_Storage_Interface.h, 58
- MsgFwStoCreateHandle
 - Msg_Framework_Storage_Interface.h, 78
- MsgFwStoDelAllMsgInFolder
 - Msg_Framework_Storage_Interface.h, 59
- MsgFwStoDeleteAccount
 - Msg_Framework_Storage_Interface.h, 60
- MsgFwStoDeleteFolder
 - Msg_Framework_Storage_Interface.h, 60
- MsgFwStoDeleteMessage
 - Msg_Framework_Storage_Interface.h, 61
- MsgFwStoDestroyHandle
 - Msg_Framework_Storage_Interface.h, 78
- MsgFwStoDetachStorage
 - Msg_Framework_Storage_Interface.h, 62
- MsgFwStoGetAccountIdsByRootId
 - Msg_Framework_Storage_Interface.h, 62
- MsgFwStoGetAccountIdsByStorage
 - Msg_Framework_Storage_Interface.h, 63
- MsgFwStoGetCommAccount
 - Msg_Framework_Storage_Interface.h, 64
- MsgFwStoGetDefaultAccID
 - Msg_Framework_Storage_Interface.h, 64
- MsgFwStoGetFolder
 - Msg_Framework_Storage_Interface.h, 65
- MsgFwStoGetFolderByType
 - Msg_Framework_Storage_Interface.h, 65
- MsgFwStoGetFolderIdsByStorage
 - Msg_Framework_Storage_Interface.h, 66
- MsgFwStoGetFolderParentId
 - Msg_Framework_Storage_Interface.h, 67
- MsgFwStoGetMessageIdsByStorage
 - Msg_Framework_Storage_Interface.h, 67
- MsgFwStoGetMessageParentId
 - Msg_Framework_Storage_Interface.h, 68
- MsgFwStoGetMsgCommInfo
 - Msg_Framework_Storage_Interface.h, 68
- MsgFwStoGetStorageStatus
 - Msg_Framework_Storage_Interface.h, 69
- MsgFwStoInitStorage
 - Msg_Framework_Storage_Interface.h, 70
- MsgFwStoQueryAccount
 - Msg_Framework_Storage_Interface.h, 70
- MsgFwStoQueryFolder
 - Msg_Framework_Storage_Interface.h, 71
- MsgFwStoQueryMessage
 - Msg_Framework_Storage_Interface.h, 72
- MsgFwStoReleaseAccountIdList
 - Msg_Framework_Storage_Interface.h, 73
- MsgFwStoReleaseCommAccount
 - Msg_Framework_Storage_Interface.h, 73
- MsgFwStoReleaseFolder
 - Msg_Framework_Storage_Interface.h, 73
- MsgFwStoReleaseFolderIdList
 - Msg_Framework_Storage_Interface.h, 74
- MsgFwStoReleaseMessageList
 - Msg_Framework_Storage_Interface.h, 74
- MsgFwStoReleaseMsgCommon
 - Msg_Framework_Storage_Interface.h, 74
- MsgFwStoSetDefaultAccID
 - Msg_Framework_Storage_Interface.h, 75
- MsgFwStoUpdateAccount
 - Msg_Framework_Storage_Interface.h, 75
- MsgFwStoUpdateFolder
 - Msg_Framework_Storage_Interface.h, 76
- MsgFwStoUpdateMessage
 - Msg_Framework_Storage_Interface.h, 77
- msgId
 - _MSG_MESSAGE_S, 29
 - _MSG_MESSAGE_STATUS_S, 33
- msgIds
 - _MSG_MESSAGE_ID_LIST_S, 26
- MsgPlgFilterCheckRules
 - Msg_Plugin_Filter_Interface.h, 104
- MsgPlgFilterCreateHandle
 - Msg_Plugin_Filter_Interface.h, 106
- MsgPlgFilterDestroyHandle
 - Msg_Plugin_Filter_Interface.h, 106
- MsgPlgFilterFinalize
 - Msg_Plugin_Filter_Interface.h, 105
- MsgPlgFilterInitialize
 - Msg_Plugin_Filter_Interface.h, 105
- MsgPlgStoAddAccount
 - Msg_Plugin_Storage_Interface.h, 109
- MsgPlgStoAddMessage
 - Msg_Plugin_Storage_Interface.h, 109
- MsgPlgStoAttachStorage
 - Msg_Plugin_Storage_Interface.h, 110
- MsgPlgStoCheckStatus

- Msg_Plugin_Storage_Interface.h, [111](#)
- MsgPlgStoCreateHandle
 - Msg_Plugin_Storage_Interface.h, [119](#)
- MsgPlgStoDelAllMsgInFolder
 - Msg_Plugin_Storage_Interface.h, [111](#)
- MsgPlgStoDeleteAccount
 - Msg_Plugin_Storage_Interface.h, [112](#)
- MsgPlgStoDeleteMessage
 - Msg_Plugin_Storage_Interface.h, [112](#)
- MsgPlgStoDestroyHandle
 - Msg_Plugin_Storage_Interface.h, [120](#)
- MsgPlgStoDetachStorage
 - Msg_Plugin_Storage_Interface.h, [113](#)
- MsgPlgStoGetAccount
 - Msg_Plugin_Storage_Interface.h, [114](#)
- MsgPlgStoGetMessage
 - Msg_Plugin_Storage_Interface.h, [114](#)
- MsgPlgStoGetStorageStatus
 - Msg_Plugin_Storage_Interface.h, [115](#)
- MsgPlgStoInitStorage
 - Msg_Plugin_Storage_Interface.h, [116](#)
- MsgPlgStoReleaseAccount
 - Msg_Plugin_Storage_Interface.h, [116](#)
- MsgPlgStoReleaseMessage
 - Msg_Plugin_Storage_Interface.h, [117](#)
- MsgPlgStoSetFwStoHandle
 - Msg_Plugin_Storage_Interface.h, [117](#)
- MsgPlgStoUpdateAccount
 - Msg_Plugin_Storage_Interface.h, [118](#)
- MsgPlgStoUpdateMessage
 - Msg_Plugin_Storage_Interface.h, [118](#)
- MsgPlgTransCancelRequest
 - Msg_Plugin_Transport_Interface.h, [122](#)
- MsgPlgTransCreateHandle
 - Msg_Plugin_Transport_Interface.h, [126](#)
- MsgPlgTransDestroyHandle
 - Msg_Plugin_Transport_Interface.h, [126](#)
- MsgPlgTransDiscardMsg
 - Msg_Plugin_Transport_Interface.h, [122](#)
- MsgPlgTransFinalize
 - Msg_Plugin_Transport_Interface.h, [123](#)
- MsgPlgTransInitialize
 - Msg_Plugin_Transport_Interface.h, [123](#)
- MsgPlgTransOnMsgIncoming
 - Msg_Plugin_Transport_Interface.h, [123](#)
- MsgPlgTransOnStatusChanged
 - Msg_Plugin_Transport_Interface.h, [124](#)
- MsgPlgTransRegisterListener
 - Msg_Plugin_Transport_Interface.h, [125](#)
- MsgPlgTransSubmitRequest
 - Msg_Plugin_Transport_Interface.h, [125](#)
- msgType
 - [_MSG_ACCOUNT_S, 10](#)
 - [_MSG_MESSAGE_REQUEST_S, 27](#)
 - [_MSG_MESSAGE_S, 29](#)
- pAddAccount
 - [_MSG_FW_STORAGE_S, 24](#)
 - [_MSG_PLG_STORAGE_S, 38](#)
- pAddFolder
 - [_MSG_FW_STORAGE_S, 24](#)
- pAddMsg
 - [_MSG_FW_STORAGE_S, 24](#)
 - [_MSG_PLG_STORAGE_S, 38](#)
- parentId
 - [_MSG_FOLDER_S, 20](#)
 - [_MSG_MESSAGE_S, 29](#)
- pAttachStorage
 - [_MSG_FW_STORAGE_S, 24](#)
 - [_MSG_PLG_STORAGE_S, 38](#)
- pCheckStatus
 - [_MSG_FW_STORAGE_S, 24](#)
 - [_MSG_PLG_STORAGE_S, 38](#)
- pCommExtData
 - [_MSG_FOLDER_S, 21](#)
 - [_MSG_MESSAGE_S, 31](#)
- pContacts
 - [_MSG_CONTACT_LIST_S, 12](#)
- pCountAccount
 - [_MSG_FW_STORAGE_S, 24](#)
- pCountMsg
 - [_MSG_FW_STORAGE_S, 24](#)
- pCountSubFolder
 - [_MSG_FW_STORAGE_S, 24](#)
- pCountUnreadMsg
 - [_MSG_FW_STORAGE_S, 24](#)
- pDelAllMsg
 - [_MSG_PLG_STORAGE_S, 38](#)
- pDeleteAccount
 - [_MSG_FW_STORAGE_S, 24](#)
 - [_MSG_PLG_STORAGE_S, 38](#)
- pDeleteAllMsg
 - [_MSG_FW_STORAGE_S, 24](#)
- pDeleteFolder
 - [_MSG_FW_STORAGE_S, 24](#)
- pDeleteMsg
 - [_MSG_FW_STORAGE_S, 24](#)
 - [_MSG_PLG_STORAGE_S, 38](#)
- pDetachStorage
 - [_MSG_FW_STORAGE_S, 24](#)
 - [_MSG_PLG_STORAGE_S, 38](#)
- pExtData
 - [_MSG_ACCOUNT_S, 11](#)
 - [_MSG_CONTACT_S, 13](#)
 - [_MSG_MESSAGE_REQUEST_S, 28](#)
 - [_MSG_MESSAGE_S, 31](#)
 - [_MSG_MESSAGE_STATUS_S, 34](#)
- pfCancelRequest

- [_MSG_PLUGIN_TRANSPORT_S](#), 40
- [pfCheckRules](#)
 - [_MSG_PLUGIN_FILTER_MANAGER_S](#), 39
- [pfDiscardMsg](#)
 - [_MSG_PLUGIN_TRANSPORT_S](#), 40
- [pfFinalize](#)
 - [_MSG_PLUGIN_FILTER_MANAGER_S](#), 39
 - [_MSG_PLUGIN_TRANSPORT_S](#), 40
- [pfInitialize](#)
 - [_MSG_PLUGIN_TRANSPORT_S](#), 40
- [pfInititalize](#)
 - [_MSG_PLUGIN_FILTER_MANAGER_S](#), 39
- [pfReceiveMsgCallback](#)
 - [_MSG_TRANSPORT_LISTENER_S](#), 48
- [pfRegisterListener](#)
 - [_MSG_PLUGIN_TRANSPORT_S](#), 40
- [pfStatusChangedCallback](#)
 - [_MSG_TRANSPORT_LISTENER_S](#), 48
- [pfSubmitRequest](#)
 - [_MSG_PLUGIN_TRANSPORT_S](#), 40
- [pGetAccount](#)
 - [_MSG_PLG_STORAGE_S](#), 38
- [pGetAccountIdsByRootId](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetAccountIdsByStorage](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetComAccount](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetDefaultAccID](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetFolder](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetFolderByType](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetFolderIdsByStorage](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetFolderParentId](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetMessageIdsByStorage](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetMsg](#)
 - [_MSG_PLG_STORAGE_S](#), 38
- [pGetMsgCommonInfo](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetMsgParentId](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pGetStorageStatus](#)
 - [_MSG_FW_STORAGE_S](#), 24
 - [_MSG_PLG_STORAGE_S](#), 38
- [pInitStorage](#)
 - [_MSG_FW_STORAGE_S](#), 24
 - [_MSG_PLG_STORAGE_S](#), 38
- [pMsg](#)
 - [_MSG_MESSAGE_REQUEST_S](#), 27
- [pMsgTypes](#)
 - [_MSG_MESSAGE_TYPE_LIST_S](#), 35
- [pQueryAccount](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pQueryFolder](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pQueryMsg](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pReleaseAccount](#)
 - [_MSG_PLG_STORAGE_S](#), 38
- [pReleaseAccountIdList](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pReleaseComAccount](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pReleaseFolder](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pReleaseFolderIdList](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pReleaseMessageIdList](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pReleaseMsg](#)
 - [_MSG_PLG_STORAGE_S](#), 38
- [pReleaseMsgCommon](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [priority](#)
 - [_MSG_MESSAGE_S](#), 31
- [pSetDefaultAccID](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pSetFwStoHandle](#)
 - [_MSG_PLG_STORAGE_S](#), 38
- [pStorageIds](#)
 - [_MSG_STORAGE_ID_LIST_S](#), 46
- [pString](#)
 - [_MSG_STRING_S](#), 47
- [pUpdateAccount](#)
 - [_MSG_FW_STORAGE_S](#), 24
 - [_MSG_PLG_STORAGE_S](#), 38
- [pUpdateFolder](#)
 - [_MSG_FW_STORAGE_S](#), 24
- [pUpdateMsg](#)
 - [_MSG_FW_STORAGE_S](#), 24
 - [_MSG_PLG_STORAGE_S](#), 38
- [queries](#)
 - [_MSG_QUERY_LIST_S](#), 42
- [queryId](#)
 - [_MSG_QUERY_S](#), 43
- [queryIds](#)
 - [_MSG_QUERY_ID_LIST_S](#), 41
- [queryValue](#)
 - [_MSG_QUERY_S](#), 43
- [reqId](#)
 - [_MSG_MESSAGE_REQUEST_S](#), 27

- [_MSG_MESSAGE_STATUS_S, 33](#)
- reqType
 - [_MSG_MESSAGE_REQUEST_S, 27](#)
- rootFolderId
 - [_MSG_ACCOUNT_S, 10](#)
- size
 - [_MSG_MESSAGE_S, 31](#)
- sortType
 - [_MSG_FOLDER_S, 20](#)
 - [_MSG_SORT_RULE_S, 44](#)
- sortTypes
 - [_MSG_SORT_TYPE_LIST_S, 45](#)
- status
 - [_MSG_MESSAGE_S, 31](#)
 - [_MSG_MESSAGE_STATUS_S, 33](#)
- storageId
 - [_MSG_ACCOUNT_S, 10](#)
 - [_MSG_FOLDER_S, 20](#)
 - [_MSG_MESSAGE_S, 30](#)
- subject
 - [_MSG_MESSAGE_S, 30](#)
- subType
 - [_MSG_MESSAGE_TYPE_S, 36](#)
- to
 - [_MSG_MESSAGE_S, 30](#)
- true
 - [Msg_Mapi_Types.h, 96](#)
- valueLength
 - [_MSG_FILTER_S, 18](#)
 - [_MSG_QUERY_S, 43](#)